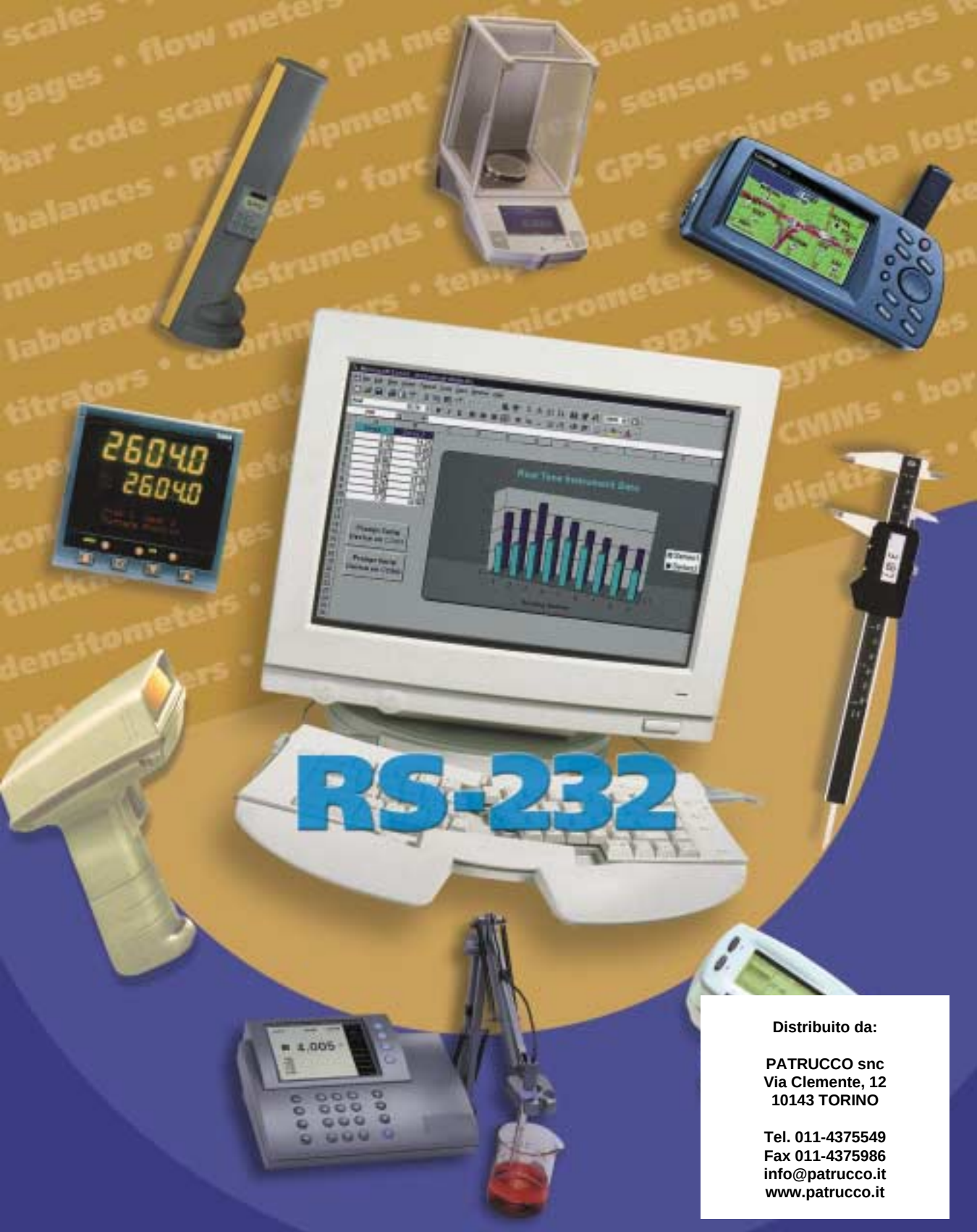


Distribuito da:

PATRUCCO snc
Via Clemente, 12
10143 TORINO

Tel. 011-4375549
Fax 011-4375986
info@patrucco.it
www.patrucco.it

WinWedge®

SoftwareWedge for Windows

Connect any instrument to your PC and instantly input real-time serial data into any application.

Version 3.0
Professional Edition

License Agreement: 2
What is the Software Wedge? 3
What can you do with the Software Wedge? 3
Before You Begin 4
System Requirements 4
Devices Compatible with the Software Wedge 4
What's New In Version 3.0 5
Why Is It Called the Software Wedge? 5
Who's Using The Software Wedge 6
Getting Started 7
Installing and Running the Software Wedge 7
How To Obtain Technical Support 7
Quick Start Guide To Using The Software Wedge 8
A Typical Software Wedge Configuration Example 9
The Main Menu 12
The File Menu 12
Software Wedge Configuration Files 13
Activating The Wedge Automatically With a Specific Configuration File 14
The Mode Menu 15
Send Keystrokes Mode 15
DDE Server Mode 17
Sending Keystrokes vs. Dynamic Data Exchange 18
Log To Disk Mode 19
The Port Menu 20
The Port Settings Dialog Box 21
The Port Analyze Menu Option 22
The Define Menu 24
Input Data Record Structure 24
Pre-Input Character Translations 24
Pre-Transfer Character Translation Table 24
Serial Output Strings 25
Hot Keys and Hot Key Actions 25

Command Source DDE Link 25
Output Buffer DDE Link 25
Defining the Input Data Record Structure 26
The Start Of Record Event 26
The End Of Record Event....27
Single field data records 29
Multiple delimited data fields 30
Special Considerations Regarding Delimiter Characters 31
Multiple fixed length data fields 31
Specifying Filters, Output Formats, and Field Lengths 32
Selecting Filters 33
Specifying Math Expressions....34
Defining Output Formats 34
Specifying Field Lengths 35
Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode) 35
Specifying DDE commands (DDE Server Mode) 36
Specifying Pre / Postambles (Log To Disk Mode) 38
Translation Tables....39
Pre-Input Character Translation Table 39
Pre-Transfer Character Translation Table 41
Defining Serial Output Strings....43
Defining Hot Keys and Hot Key Actions....45
Selecting Hot Key Actions 45
Selecting Hot Key Keystrokes 46
Defining A Command Source DDE Link 47
Defining An Output Buffer DDE Link 48
The Activate Menu 49
Activating the Wedge in Test Mode or Normal Mode....51
The Software Wedge Window 51
The Edit Menu 52
The Quit Menu 54
Activating the Software Wedge in Virtual Instrument Mode 55
Accessing On-Line Help....56

Keystroke Macro Rules 57
Date And Time Stamps In Macros 58
Format Expressions 59
Formatting Numbers 59
Formatting Character Strings 64
Formatting Date/Time Expressions 65
Mathematical Functions and Operators 69
Operator Precedence in Math Expressions 71
Understanding Dynamic Data Exchange 72
Establishing DDE Links Using The Windows Clipboard 73
DDE and the Software Wedge 74
Software Wedge DDE Commands 75
Using the LINKTEST Utility to test DDE Commands 77
Tech Notes 78
Fixing a problem in the Windows DDEML.DLL Dynamic Link Library 78
Understanding How DDE Works Helps Avoid Common Programming Mistakes 79
DDE Examples 80
Important: 80
DDE Examples for Excel 5 - 2000 (Visual Basic for Applications) 81
DDE Examples for Microsoft Access 96
DDE Examples for Microsoft Word for Windows 98
DDE Examples for Visual FoxPro for Windows 99
DDE Examples for Microsoft Visual Basic for Windows 102
DDE Example for Lotus 123 for Windows 103
DDE Examples for Quattro Pro for Windows 104
Passing Data Across a Network Using Network DDE 106
Diagnosing Serial Communications Problems 108
Troubleshooting 109
Running The Wedge On Multiple Serial Ports Simultaneously 111
Using More Than Two Serial Adapters At A Time 111
Understanding The Data From Your Serial Device 112
More Software Wedge Configuration Examples 116

A More Complex Example 119
Cool Wedge Tricks 121
FileWedge 130
Differences Between FileWedge and The Software Wedge 130
TCP/Wedge (WinWedge Pro 32 Bit Edition Only) 132
Differences Between TCP/Wedge and The Software Wedge 132
INTRODUCTION TO SERIAL COMMUNICATIONS 134
Bi-directional Communications 134
Synchronous And Asynchronous Communications 134
Communicating By Bits 135
The Parity Bit 135
Baud Versus Bits Per Second 135
RS-232C 136
DCE And DTE Devices 136
Cable Lengths 138
Cables, Null Modems, And Gender Changers 138
Null Modem Cables and Null Modem Adapters 139
9 Pin To 25 Pin Adapters 139
Gender Changers 139

License Agreement:

1. GRANT OF LICENSE

TAL Technologies, Inc. grants you the right to use one copy of the enclosed software program (the SOFTWARE) on a single terminal connected to a single computer (i.e., with a single CPU). You may not network the SOFTWARE or otherwise use it on more than one computer or terminal at the same time.

2. COPYRIGHT

The SOFTWARE is owned by TAL Technologies, Inc. and is protected by United States copyright laws and treaties. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the users manual accompanying the SOFTWARE.

3. OTHER RESTRICTIONS

You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, de-compile, or disassemble the SOFTWARE. If SOFTWARE is an update, any transfer must include the update and all prior versions.

4. DISCLAIMER

No Warranty of any kind on the SOFTWARE is expressed or implied.

In no event shall TAL Technologies or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profit, business interruption, loss of business information or other pecuniary loss) arising out of the use of or inability to use this product. If you have questions concerning this agreement, or wish to contact TAL Technologies, please write:

TAL Technologies, Inc.
2027 Wallace Street
Philadelphia, PA 19130 USA
Tel: (215)-763-5096 Fax: (215)-763-9711
email: sales@taltech.com
Web Site: http://www.taltech.com

What is the Software Wedge?

The Software Wedge is an extremely powerful utility that is designed to add serial communications capabilities to any Windows 3.X, 95/98/NT or 2000 application. The main function of the Wedge is to provide a way to connect any RS232, RS422 or RS485 device to a PC and be able to read or write data to or from the device directly from within any Windows application without the need for a custom serial device driver. Typical kinds of serial instruments being used with the Software Wedge include bar code and magnetic stripe readers, electronic balances, PLCs, measuring tools including calipers and gages, temperature sensors, telephone call logging systems, pH and color meters, coordinate measuring machines, optical comparators, digital volt meters, GPS receivers and a wide variety of other laboratory or industrial instruments that communicate over a standard serial interface.

The Software Wedge offers an array of features including the ability to parse and filter data received through the serial port as well as add additional data including date/time stamps or cursor navigation keystrokes to incoming data. Two translation tables allows you to translate incoming characters to other characters or to specific keystrokes. You may even pre-define "Serial Output Strings" that can be transmitted to your serial device either automatically at regular timed intervals or by clicking your mouse on a button in the Software Wedge window or by pressing a “Hot Key” on your PC’s keypad.

The Software Wedge can transfer serial data to *any* Windows application either by sending the data as keystrokes or by providing data as a DDE (dynamic data exchange) server. You can even set up the “Wedge” to issue commands to another application after each record of incoming serial data has been transferred to it. Thus you could cause the receiving application to perform actions based on the data being received; for example you could force an application to run a macro or update a graph or chart in real time using the data received from the device. To send data out the serial port, you can use either a special set of DDE commands to transmit character strings or you can even place a serial output buffer directly in the application that you want to communicate with. The Wedge can also be used to capture serial data directly to a disk file while you work with other programs on your PC. All these features make it possible to create extremely sophisticated device control interfaces for virtually any serial instrument directly from within any Windows application program.

What can you do with the Software Wedge?

The Software Wedge can be used in applications ranging from simple data input (i.e. reading data from a bar code scanner into a database) to complex device control interfaces (i.e. using the Wedge with Excel, InTouch, The FIX, or other HMIs or LIMS to create an On-Line, real time, process control program that communicates with several data collection devices). The best way to understand the complete capabilities of the Wedge is to read this manual thoroughly. The Software Wedge has many features including powerful DDE capabilities that are not immediately apparent from the menus and dialog boxes presented when you run the Wedge on your PC. Again, the only way to learn about its full capabilities is to read this manual in its entirety.

See Also: Cool Wedge Tricks

Before You Begin

We at TAL Technologies would like to take this opportunity to thank you for purchasing the Software Wedge. We sincerely hope that it proves to be the perfect solution for all your serial I/O applications. We also hope that you find the Software Wedge both easy and enjoyable to use. If there is a particular feature that you would like to see in a future version or perhaps a change in either the software or the documentation that would make this product easier to use, please contact us. Your feedback will be greatly appreciated.
We would also like to take this opportunity to remind you to read the copyright notice and license agreement on page two of this manual thoroughly. You have purchased a license to use this software in a single PC and thus you may not use it on more than one PC at a time without purchasing additional licenses. Please help us to protect your investment by adhering to the terms of this agreement and by reporting copyright violations directly to: TAL Technologies, Inc.
Tel: 800-722-6004 or 215-763-5096 Fax: 215-763-9711 email: support@taltech.com

System Requirements

The 16 bit version of the WinWedge will run on any IBM or compatible PC running MS Windows 3.x, Windows 95/98 or Windows NT / 2000. The 32 bit version of WinWedge will run on any Windows 95, 98 or Windows NT or 2000 system. The Wedge uses roughly 2 Mb of disk space and 4 or more Mb of RAM memory is recommended.

If you are running Windows 3.1, it is also recommended that you have the TERMINAL font installed in your system. The Terminal font is used in the Software Wedge to display control codes that might appear in data received from your serial device. The Software Wedge will function properly without the Terminal font however you may find it easier to configure the Wedge with this font installed. The Terminal font is shipped with Windows and can usually be found in your "\Windows\System" directory. You can check for it or you can install it using the "Font" applet in the Control Panel located in the MAIN program group in the Program Manager. For instructions on using the Font applet, refer to your Windows users manual. All 32 bit versions of Windows have the Terminal font built into the operating system therefore there is no need to install it.

Devices Compatible with the Software Wedge

The Software Wedge was designed primarily as an interface to most commonly used RS232 data collection devices including Electronic Scales, Bar Code and Magnetic Stripe Readers, Electronic Measuring Tools, Medical, Industrial and Laboratory Instruments, Telephone PBX and Caller ID systems, Data Loggers, Flow Meters, Sensors, Plate Readers, Spectrum Analyzers, VOMs, pH meters, Densitometers, CMMs, Strip Chart Recorders, Titrators, Quality and Process Control Instruments, Coin Counters, PLCs, GPS receivers etc.. Any type of device with a serial interface whose output data is transmitted in a somewhat structured ASCII format is a good candidate for use with the Wedge. WinWedge Pro is extremely flexible and can even be used with many devices that transmit Hexadecimal or even Binary data.

Note: Many RS422 and RS485 serial devices can also be used with the Wedge provided that you have the correct type of serial I/O adapter or converter installed in your PC.

What's New In Version 3.0

Version 3.0 of The Software Wedge adds several new capabilities to previous versions including: math functions, user definable “Hot Keys” that allow you to control many serial I/O functions with a simple key press, support for up to 9 serial ports under Windows 3.x or up to 99 serial ports under Windows 95/98, NT or 2000 (32 bit version only), a Pre-Input Character Translation Table, a Disk File Logging Mode, additional Data Parsing and Filtering capabilities as well as Data Formatting, a Timed Automatic Serial Output Function and an expanded set of DDE Commands. Complete context sensitive help has also been added to all windows and dialog boxes thus making the Wedge easier to set up and use. A new “Virtual Instrument” mode even allows you to test the functionality of the Wedge without requiring an actual serial instrument to be attached to your PC. The users manual has also been re-written to include many examples for setting up and using the Wedge with different serial devices and other software products. All new features were designed to support a wider range of RS232 instruments and I/O data formats as well as make the Software Wedge much easier to set up and use.
The 32 bit Professional Edition of WinWedge also includes TCPWedge which is a version of the Software Wedge designed for data collection across a TCP/IP network.

Why Is It Called the *Software Wedge*?

A *Wedge* (or *keyboard wedge*) is a hardware device that connects between a computer and its keyboard. Typically a wedge provides a serial port that allows data to be inputted from a serial device as if it was being typed in on the keyboard.

Before the development of the *Software Wedge*, keyboard wedges were the only available solution if you needed to input serial data directly into an off the shelf program like Excel, Lotus or dBase, etc. Unfortunately, keyboard wedges are severely limited in capability, are painfully slow, and they cannot support two way serial I/O. They are also expensive, prone to failure and can be difficult or impossible to install, especially on a laptop that does not have a keyboard port.

The Software Wedge is a *software only* alternative to a keyboard wedge that allows you to connect a serial device directly to your PC's serial port with no need for any additional hardware, thus the name: *Software Wedge.*

Because the Software Wedge is implemented as a software product, it works directly with your PC’s operating system and can therefore operate much faster than a hardware wedge. It can also provide features that are either difficult or impossible to implement in a hardware product such as data parsing, filtering, formatting and translation functions, fully bi-directional serial I/O capability, date/time stamping functions and support for operating system specific features like Dynamic Data Exchange and true background operation. If you compare features, the Software Wedge is light years ahead of even the most sophisticated hardware wedge available. If you compare prices you’ll also find the Software Wedge to be far more economical as well.

Who's Using The Software Wedge

The Software Wedge was originally developed by TAL Technologies in 1989 as a simple interface for bar code readers. It quickly became apparent that a product of this type was needed for use with the hundreds of different types of devices that communicate through a serial interface. Since then, the Software Wedge has evolved into an extremely powerful product capable of interfacing virtually any type of serial instrument to a PC. Many thousands of copies of the "Wedge" have been licensed to users in every country around the globe. The following is just a short list of some of the many companies that rely on the Wedge for hundreds of different serial data collection applications.

3M, Abbott Labs, Allen-Bradley, Allied Signal, American Cyanamid, American Express, Ametek, Amgen, Amtrack, Amway, Anderson Consulting, ARCO Chemical, Ashland Chemical, AST Research, AT& T, Atlantic Electric, Avery, BASF Corp., Battelle, Bauer Industries, Baxter Healthcare, Becton Dickinson, Bell Atlantic, Bell Industries., Black & Decker, Boeing, Borden Chemical, Bose, Bristol-Myers-Squibb, Bridgestone Firestone, Bush Industries, Butterball Turkey, Cable & Wireless, Cambridge Aeroflo, Camcar Textron, Canon USA, Catepillar, Centers for Disease Control, CERN, Chevron Research, Ciba Corp., Ciba Corning Diagnostics, Corning Glass, Cray Research, Cryovac, Dana Corp, Danfloss Electronics, Data Electronics, DataVision, Delco Chassis, Delco Remy, DHHS, DHL International, Domino Sugar, Donnelly, Dow Chemical, Dow Deutschland, Dow Jones, Dow Pipeline, Dupont, Duracell, Eaton Corp., EDS, Elan Engineering, Eli Lilly, Epson, Esselte Meto, Eurotherm Chessell, Eveready, FDA, Federal Express, Federal Reserve, Firestone, First Brands, Fischer Technologies, Fisher Controls, Fisher Rosemount Systems, Florida Power, Florida Steel, Ford Motor Co., Friskies, Frito-Lay, Fuji-Xerox, Fujitsu Microelectronics, GSA, Gates Rubber, GE Famuc, GE Medical, General Electric, General Dynamics, General Mills, General Motors, Georgia Pacific, Gillette, Goodyear, Great Lakes Controls, Grumman, Gulf States Steel, Harley Davidson, Harp Ireland, Helene Curtis, Henry Ford Hospital, Hewlett Packard, Hoechst Celanese, Hoffer Scientific, Honeywell, Hoover, Hughes Aircraft, Hunter Industries, ITI Electronics, IBM Corp., Imperial Oil, Intel, Intellution, Intermec, J.C. Penney, Jack & Jill Ice Cream, James River Corp., Jandel Scientific, Jenco Instruments, Jensen Chevron, Johnson & Johnson, Johnson Controls, Kaiser Aluminum, Kal Can Foods, Kimberly-Clark, Koch Industries, Kodak, Konica, Kraft Foods, Lake Pharmaceutical, Land O' Lakes, Leeds & Northrup, Lennox Industries, Levi Strauss, Liquid Container, Lockheed Martin, Lubrizol, Lucas Aerospace, M.I.T., M & M Mars, Malaysian Carbon, Marion Merrell Dow, Mars Electronics, Marsam pharmaceuticals, Martel Electronics, Maytag, Merck & Co., Metrologic, Moore Research, Motorola, NASA, National Gypsum, National Institute of Health, National Weather Service, Navel Medical Research, New York Times, Nestle, Nikon, Nippon Denso, NordicTrack, Northeast Utilities, Northern States Power, Northern Telecom, Norton Chemical, Nova Scotia Power, Nynex, Ocean Spray, Olympic Controls, Osram Sylvania, Ortho-McNeil Pharmaceuticals, Otter Tail Power Co., Owens Corning, PCC Airfoils, Pacific Gas and Electric, Packard Electric, Penzoil, Pfizer, Philips Consumer Electronics, Picker International, Pioneer Hi-Bred International, Pioneer Micro Systems, Pitney-Bowes, Plant Genetic Systems, Polaroid, Porter Instruments, PPG Industries, Prince Sports Group, Pratt & Whitney, Proctor & Gamble Pharmaceutical, Ragu Foods, Rand Mining, Ravenswood Aluminum, Raychem, Reynolds Metals, Richmomd Memorial Hospital, Robert Bosch Ind., Rosemount Analytical, Rockwell International, Rohr, RR Donnelley, RW Johnson Pharmaceutical Research, Sandoz, Sanyo Electric Co, Sartorius Corp., Seagate Tech., Scott Paper, Sears & Robuck, Siemens, Smith Corona, SmithKline Beecham, Spaulding, Spectra Physics, SSI Services, Stanley Tools, Sony, Southern Research Institute, Starcraft Aerospace, Strathmore Paper, Summitt Controls, Sunkist, Symbol Technologies, TCBY Ent., TEC, Teledyne, Telxon Corp., Texas Beef, Texas Instruments, Texas Medical Center, The American Tobacco Co., The Coca Cola Co., The Gap, The Glidden Co., The Mennen Co., The Pister Group, The Trane Co., Timberline Instruments, Toshiba, Toyota America, Trimos Sylvac, TVA, Unisys, Upjohn, United Parcel Service, U.K. Ministry of Defense, U.S. Telecom, U.S. Army, Navy, Coast Guard, U.S. Fish & Wildlife, U.S. Geological Survey, U.S. Gypsum, U.S. Postal Service, Underwriter's Labs, Unilever, VA Medical Center, Verifone, Verbatim, Volvo, W.L. Gore, Washington Post, Wausau, Westinghouse, Wyeth Ayerst, Xerox, Zeneca Pharmaceutical, Zymark, and many, many more....

National laboratories using the Wedge include Oak Ridge, Sandia, Los Alamos, Argonne, Fermilab and Lawrence Berkeley laboratories and hundreds of industrial and university research laboratories worldwide from Harvard and MIT to Universitat Stuttgart.

Getting Started

Installing and Running the Software Wedge

The Software Wedge comes with an installation program called "SETUP.EXE" that will automatically copy all Software Wedge program files to your hard disk and also automatically install the Software Wedge icons into the Windows 3.x Program Manager so that it can be run by simply clicking on its icon from within the Program Manager. If you are running Windows 95/98, NT or 2000, the Setup program will automatically add the Software Wedge program files to the Windows "Start Menu".

To install the Wedge, place the Software Wedge distribution diskette into drive A: (or B:) then open the Windows Program Manager and select "Run" from the Program Manager "File" menu. In Windows 95/98 or NT, select "Run" from the Start Menu. When the "Run" dialog box appears, type the command "A:SETUP" and click "OK". The SETUP program takes approximately two minutes to copy all files and install the Wedge into the Windows Program Manager or Start Menu.

The Software Wedge can be run by either double clicking the mouse cursor on the Software Wedge icon or it can also be run from the Windows Program Manager "File" & "Run" dialog box using the command "WinWedge". You could also launch the Software Wedge from any Windows program that supports the ability to launch another Windows program.

See Also: Activating The Wedge Automatically With a Specific Configuration File

How To Obtain Technical Support

If you experience difficulty configuring the Software Wedge after you have read this manual thoroughly, please call TAL Technologies at Tel: (215)-763-5096 or Fax: (215)-763-9711 for technical support. Questions can also be sent via **e-mail** to: **support@taltech.com**

Only registered users of the Software Wedge are eligible for technical support. Please have your original Software Wedge diskette(s) and this users manual available when calling for support and, if possible, be at your PC with the Software Wedge running when you call. TAL Technologies also has a web page at: **http://www.taltech.com** where you can find technical information and examples as well as a Software Wedge "Knowledge Base" containing answers to many frequently asked questions.

Important: If you have not already done so, please take the time to fill out the Software Wedge "Owner Registration Card" that came with this product. You can either mail the registration card in or you can fax it to the fax number on the card. If you call for technical support and you are not a registered user then we will not be able to provide you with telephone or e-mail support.

See Also: Diagnosing Serial Communications Problems
Troubleshooting

Quick Start Guide To Using The Software Wedge

Outlined below are the most typical minimal steps that are required to use the Wedge successfully. Please note: This section has been provided as a general overview of the typical steps involved in setting up the Software Wedge and is by no means a replacement for the rest of this manual.

Step 1. Select how the Wedge will transfer incoming serial data to your other Windows programs by choosing one of the options in the *Mode* menu. The Wedge can transfer incoming serial data to other programs by either converting the data to "keystrokes" and typing the data into another application's window or it can pass the data using Dynamic Data Exchange (DDE). Sending data as keystrokes is generally much simpler than using DDE however DDE provides greater flexibility in how you can deal with the data in the other program. For a complete description of each of the three modes available in the Mode menu, read the section of this manual entitled: **The Mode Menu**

Step 2. Select the *Settings* option in the *Port* menu and choose the communications parameters that match the settings for the serial device that you will be using. After you select your communications parameters, you should select the *Analyze* option from the *Port* menu and transmit some sample data from your serial device in order to both test that you are using the correct serial communications parameters and also to gain an understanding of how the data from your serial device is structured.Refer to the sections: **The Port Menu** and **Diagnosing Serial Communications Problems** for further information.

Step 3. Select "*Input Data Record Structure..."* from the *Define* menu. This will present you with a series of dialog boxes that allow you to first describe the structure of your incoming serial data and finally describe how you want to parse, filter and format the data as well as add additional data, keystrokes, or DDE commands to the data before passing it on to the target application program. This step can be fairly complex therefore it is highly recommended that you read the sections: **Defining the Input Data Record Structure** and **Understanding The Data From Your Serial Device**

Step 4. If the device that you are using requires data or commands to be sent back to it from your PC, please read the sections: **Defining Serial Output Strings** and **Defining Hot Keys and Hot Key Actions**

Step 5. The final step is to Activate the Wedge to start it working. You may want to save your configuration at this point by selecting "*Save*" from the *File* menu. To activate the Wedge simply select "*Test Mode*" from the *Activate* menu.
Refer to the section: **The Activate Menu** for further information.

Note: The Wedge will not work until you activate it by selecting one of the options in the Activate menu. We recommend selecting "Test Mode" until you have everything working the way you desire.

See Also:
A Typical Software Wedge Configuration Example
More Software Wedge Configuration Examples
Diagnosing Serial Communications Problems
Troubleshooting

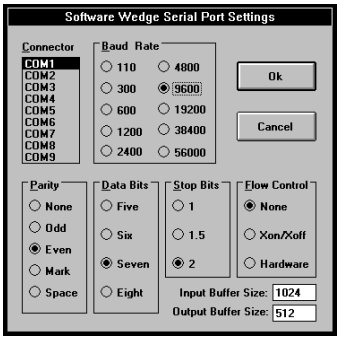
A Typical Software Wedge Configuration Example

For Bar Code Readers and Simple Measuring Instruments

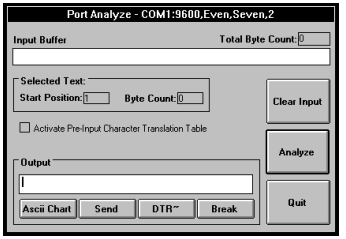
This section shows the typical steps for setting up the Wedge for use with a bar code reader or a simple measuring instrument like a digital caliper, force gage, weigh scale, or pH meter. The steps outlined below show how to configure the Wedge to read data from your device directly into another Windows application wherever the cursor is; as if the data were being typed in on your keypad. This example assumes that the application that you primarily want to capture data to is a spreadsheet. It also assumes that you would like to have successive readings from your instrument stacked up in a column in the spreadsheet.

You begin by reading the owners manual for your device and find that it uses the serial communications parameters of 9600 baud, even parity, 7 data bits and two stop bits.

Step 1. Plug the instrument into serial port COM1 or COM2, turn it on, and run the Software Wedge by double clicking on its icon in the Windows 3.x Program Manager or by selecting the Software Wedge from the Windows 95/98/NT "Start" menu.



Step 2. Select the “SETTINGS” option from the “PORT” menu in the Software Wedge and choose the communications parameters COM1 (or COM2), 9600 baud, Even Parity, 7 Data Bits, two stop bits and no flow control from the dialog box that appears and then click the OK button to return to the main menu.

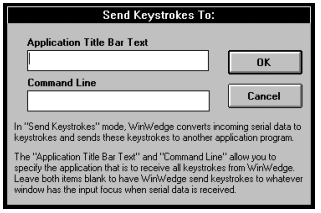


Step 3. Select “ANALYZE” from the “PORT” menu and transmit a sample reading from your device into the "Input Buffer" in the Analyze window. This step serves two purposes, first to simply test that your communications parameters are correct and that the device is connected properly and second to gain an understanding of how the data from the device is structured. If data does not appear in the text box marked “Input Buffer”, then read the section of this manual entitled “Troubleshooting” .

If data appears in the Input Buffer then examine all the characters that are received looking for regular features like delimiters, or special characters like carriage returns or linefeeds, etc..
(A carriage return (ASCII 13) will appear as a musical note and a linefeed (ASCII 10) will look like a small rectangle with a circle inside it.) You may want to take several readings from the device just to make sure that each reading is structured in a similar manner.

Most simple devices such as bar code readers, calipers, force gages, and electronic balances transmit a single number or short data string followed by a carriage return or carriage return linefeed pair. For the rest of this example, we will assume that this is how the data from your device is structured.

After you have examined the data from your device thoroughly and are satisfied that the device is transmitting your data correctly, click on the button marked “Quit” to return to the main menu.


Send Keystrokes To:
Application Title Bar Text
OK
Command Line
Cancel
In "Send Keystrokes" mode, WinWedge converts incoming serial data to keystrokes and sends these keystrokes to another application program.
The "Application Title Bar Text" and "Command Line" allow you to specify the application that is to receive all keystrokes from WinWedge. Leave both items blank to have WinWedge send keystrokes to whatever window has the input focus when serial data is received.

Step 4. Because we want to be able to read data directly into *any* Windows program, the easiest way to do this is to have the Wedge convert the serial data to keystrokes so that it will appear as if it is being typed in on your keyboard. Select “Send Keystrokes To...” from the MODE menu (even if it is already checked). This will present a dialog box prompting for the application title bar text and/or command line that will identify the application that you want the Wedge to send all keystrokes to.

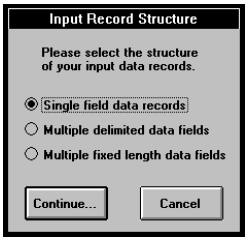
In this example, we want the Wedge to send the data to whatever application has the input focus when the data is entered therefore the two items “Application Title Bar Text” and “Command Line” should be completely cleared out. Leaving these items blank (as shown) causes the Wedge to send the data from your device as keystrokes to whatever application has the input focus when the data comes in through the serial port. After clearing out these two items, click the OK button to return to the main menu.

Step 5. The next step is to define the structure of the data that is being received from your serial device. We primarily want to send the data to a spreadsheet and have the cursor move down one cell after each reading, hence we must also instruct the Wedge to add a Down Arrow keypress at the end of each reading from our device.

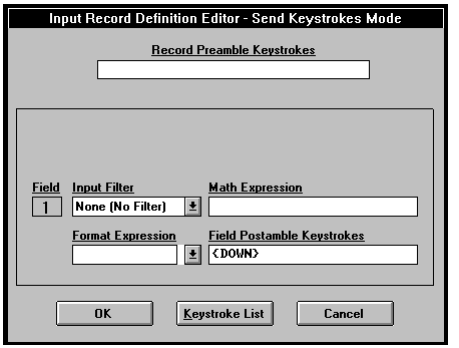
To do this, select “Input Data Record Structure...” from the DEFINE menu. The first dialog box that

appears (left) will allow us to describe the events that determine the start and end of each data record transmitted by our device. In this case each data record is a small text string ending with a carriage return (ASCII 13). Thus, we would specify "Any Character Received" as the "Start of Record Event" and “Carriage Return or CrLf Received” as the “End of Record Event”.

After these options are selected, click on the “Continue...” button to proceed

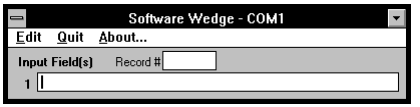


The next dialog box to appear provides options to describe an overall "Structure" for each data record. In this example our data only consists of a single "Field" therefore we would select "Single Field Data Records" and click the "Continue..." button.



In the next dialog box to appear, place the cursor in the text box labeled "Field Postamble Keystrokes" and then click the button marked "Keystroke List". When the keystroke list appears, scroll down through the list until the word "Down" is highlighted and then click the OK button in the keystroke list dialog box. This should cause the word "{DOWN} to appear in the "Field Postamble Keystrokes" text box (as shown). Then click the button marked "OK" to return to the main menu.

The Software Wedge is now completely configured to work the way we want it to. You should save your configuration at this point by selecting "Save" from the FILE menu. It is a good idea to choose names for your configuration files that are easily associated with the device that you are using. For example you might use the name BARCODE.SW3 for a bar code reader.



Step 6. After configuring the Software Wedge you must activate it by selecting "Test Mode" from the ACTIVATE menu in order to start it working. After you activate the Wedge, a window containing a text box marked "Input Data Field(s)" will appear on your screen to indicate that it has been activated successfully.

Step 7. At this point, the Wedge is now ready and waiting to do its job. If you run your spreadsheet (or any other Windows program) and place the cursor in a cell and then transmit a reading from your serial device, the data will appear just as if it were being typed in on your keyboard with a "Down Arrow" keypress being issued automatically after the data appears.

See Also: More Software Wedge Configuration Examples
Cool Wedge Tricks

The Main Menu

When you run the Software Wedge you will be presented with the following "Main Menu":

The title bar of the main menu indicates that you are in *setup* mode and it also displays the filename of any currently selected configuration file in parentheses. Before you can use the Software Wedge, you must configure it for your particular serial device. You configure the Wedge by selecting various options from the main menu and then after it is configured, you “Activate” it by selecting one of the options in the Activate menu.

The six main menu options and their functions are :

File - Allows opening and saving of configuration files and exiting of the Software Wedge. See Also: The File Menu

Mode - Allows selection of the method used to transfer serial input data from the Software Wedge to another application or disk file & optionally specify the application or disk file that is to receive the data. See Also: The Mode Menu

Port - Allows selection and testing of all serial port communications parameters. See Also: The Port Menu

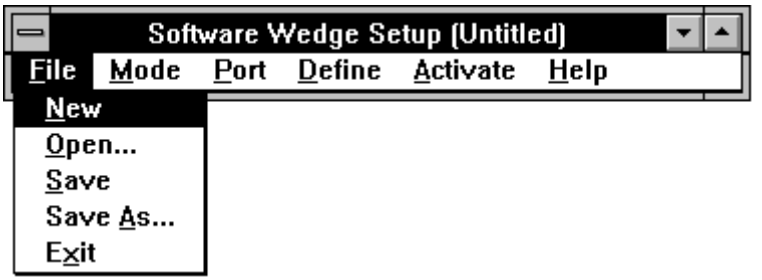
Define - The define menu contains selections that allow you to customize how the Software Wedge should work for your particular application. The different sub-menu items allow the definition of the input data record structure, parsing instructions & filters to be applied to incoming data, additional cursor navigation keystroke or DDE commands to be issued during the input of data, translations of incoming characters, definition of hot keys and serial output data strings, and the definitions of both a DDE Command source and an DDE Output Buffer source. See Also: The Define Menu

Activate - Allows activation of the Software Wedge using the current configuration. The Software Wedge will not do anything until it is activated by selecting one of the activation options in this menu. This menu also has entries that allow you to control several additional features of the Software Wedge including how it appears when activated. See Also: The Activate Menu

Help - Provides On-Line help for the Software Wedge. In addition to the help menu, the Software Wedge has complete context sensitive help that is available by pressing function key F1. See Also: Accessing On-Line Help

The File Menu

Selecting "File" from the Software Wedge main menu presents the sub-menu shown below:



File sub-menu options and their functions are as follows:

New - Unloads any currently loaded configuration file and resets the Software Wedge to its default configuration.

Open - Opens a dialog box that allows selection of a previously saved Software Wedge configuration file.

Save - Saves changes to the currently open Software Wedge configuration file.

Save As - Opens a dialog box that allows you to save the current configuration with a filename that you specify.

Exit - Exits the Software Wedge.

Software Wedge Configuration Files

When you configure the Software Wedge for use with a particular serial device and a specific application program, you can save your configuration to a disk file that can be reloaded later. By saving your configurations to a disk file, you only have to create the configuration once. The next time you need to use the Wedge with a particular device, you can simply run the Wedge and load in your previously saved configuration file and then activate the Wedge. This saves you the trouble of having to configure the Wedge each time you want to use it.

The first four options in the "FILE" menu allow you to save and load your configuration files.

Activating The Wedge Automatically With a Specific Configuration File

The Software Wedge supports one optional command line argument using the following syntax: **WinWedge.exe *filename*** where *filename* is the name of a previously saved Software Wedge configuration file.

If a filename is supplied on the command line used to launch the Wedge, the specified configuration file will be loaded and the Wedge will activate itself automatically using the configuration. The filename must be the name of a valid Software Wedge configuration file complete with the filename extension and also the drive and directory path for the file.

The filename extension ".SW3" is the default filename extension for all Software Wedge Professional Edition configuration files. The installation program for the Wedge (SETUP.EXE) automatically establishes an "Association" between the filename extension ".SW3" and the executable program "WinWedge.exe". This allows you to launch WinWedge configuration files that have the filename extension ".SW3" directly without having to go through the process of launching the Wedge, opening the configuration file and then activating the Wedge.

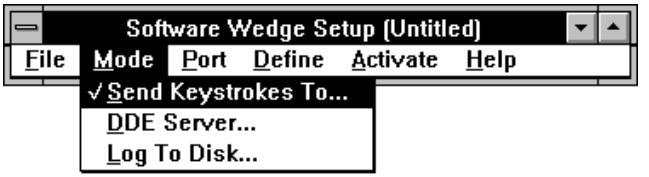
For example after you create and save a configuration file for the Wedge, you can simply double click on the name of the configuration file in the File Manager or the Windows Explorer and Windows will automatically launch the Software Wedge and activate it using the selected configuration.

Activating WinWedge Automatically When You Start Windows

To load and activate the Software Wedge automatically with a particular configuration file when you start Windows, open the Windows Explorer and select a previously saved WinWedge configuration file and drag and drop the configuration file into the folder named "StartUp". The "StartUp" folder is usually located in the "Programs" folder inside the "Start Menu" folder in your "Windows" folder.

In Windows 3.1 you can perform the same function by opening the Windows File Manager, selecting your desired WinWedge configuration file and dragging the file over to the Program Manager and dropping it into your “Startup” program group. This should cause the Software Wedge program icon to appear in your Startup group. From this point on, whenever you start Windows, the Wedge will automatically load and activate itself with the selected configuration file.

The Mode Menu



The Software Wedge can operate in one of three "modes" that control how the Wedge will transfer incoming data from your serial device to another application program or to a disk file. The Wedge can transfer data either by converting the data to keystrokes and then sending the keystrokes directly to another application or it can use Dynamic Data Exchange (DDE) to pass data to any application that supports DDE. (A simple way to see if an application supports DDE, look for a "Paste Link" or "Paste Special" command in its main menu, usually under an "Edit" menu).

The first two choices in the MODE menu, "Send Keystrokes" and "DDE Server", allow for the transfer of incoming serial data directly into another running Windows program. Each of these two modes has its own unique advantages and the choice of which mode to use depends greatly on what you are trying to accomplish as well as on the capabilities of the application program that you will be sending data to. It is extremely important that you fully understand the differences between these two modes.

See Also: Sending Keystrokes vs. Dynamic Data Exchange.

The third "Log To Disk" mode causes the Wedge to write all serial input data to a disk file instead of sending the data to another Windows application.

See Also: Log To Disk Mode.

Send Keystrokes Mode

In "Send Keystrokes Mode", the Wedge will convert incoming serial data to keystrokes and then send these keystrokes directly to any application that you specify. This is the easiest and most common way to use the Wedge with most simple devices like bar code readers, electronic balances and measuring tools like gages and meters.

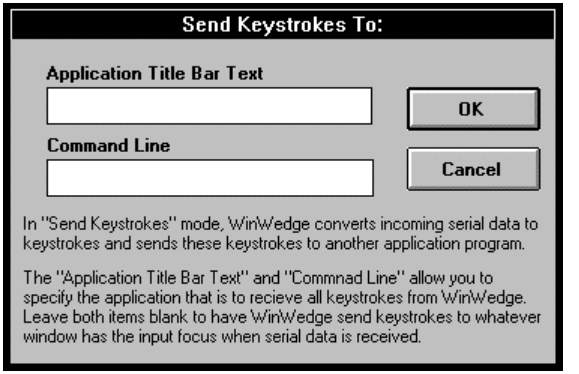
When you select "Send Keystrokes To..." from the Mode menu, a dialog box will appear that lets you specify the application program that you want the Wedge to send the data to. The Wedge is pre-configured to send keystrokes to the NOTEPAD accessory that comes with Windows however you will probably want to change this behavior. You may enter either the application's title bar text (the text that appears in the title bar of the program's main menu) or you may also specify the command line used to launch the program in Windows (the application's full pathname and EXE filename along with any command line arguments). If you specify only the title bar text, then the application must be running, either as an icon or in a window, before the Wedge can send it any data. If you specify only the application's command line, then the Wedge will automatically launch the application before sending it any data.

After you activate the Software Wedge in “Send Keystrokes” mode, whenever data is received from your serial device, the Wedge will try to send the data to the application that you specify. The logic used by the Software Wedge is to first search for an application with the specified title bar text and if the application is already running, the Wedge will activate it so that it has the input focus before sending keystrokes to it. If a window with the specified title bar text is not found or if the title bar text is not specified, then the Software Wedge will try to run the program using the command line specified. If this is successful, then the Software Wedge will retrieve the title bar text from the application after it starts running so that it can find the application’s window later on.

When specifying the title bar text, you do not need to enter the complete title and may use a substring of the text. For example, to send data to Microsoft Excel, you can simply specify "Excel" for the title bar text and as long as there are no other programs running that contain the word "Excel" in their title bar, the Wedge should be able to find Excel’s window correctly. You may specify both the title bar text and command line in either upper or lower case.

Important Note: If you do not specify a command line or a window's title bar text, the Wedge will send keystrokes to whatever application has the input focus when data is inputted from your serial device. In other words, the Wedge will work like a second keyboard. You may prefer to use the Wedge in this manner because it allows you to input data wherever the cursor happens to be in whatever application has the focus. If the Wedge itself has the focus when data is inputted, it will simply display the data without trying to send it to any other program.

Tech Tip:
Some Windows programs may have more than one window with the same Title Bar Text as the main window and therefore the Wedge may not always be able to find the right window to send keystrokes to. If you experience difficulties in getting the Wedge to send keystrokes to a particular application program even if you are sure that you specified the correct Application Title Bar Text or Command Line, try clearing out both of these items. Then after you activate the Wedge simply launch and/or switch the input focus to the application that you want to receive the data and place the cursor in the location where you want data entered.


Send Keystrokes To:
Application Title Bar Text
OK
Command Line
Cancel
In "Send Keystrokes" mode, WinWedge converts incoming serial data to keystrokes and sends these keystrokes to another application program.
The "Application Title Bar Text" and "Commnad Line" allow you to specify the application that is to recieve all keystrokes from WinWedge. Leave both items blank to have WinWedge send keystrokes to whatever window has the input focus when serial data is received.

DDE Server Mode

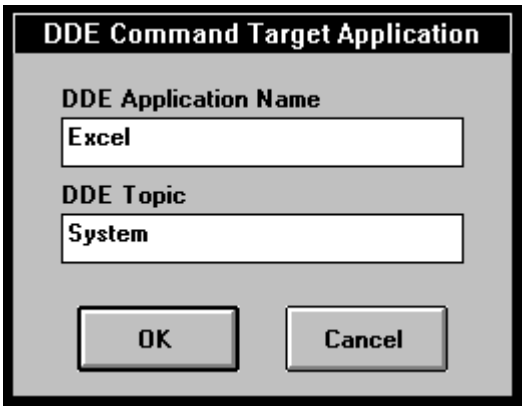
When the Software Wedge is in DDE Server mode, instead of sending incoming serial data to another application as keystrokes, the Wedge allows other Windows applications that support DDE to retrieve incoming serial data through DDE (Dynamic Data Exchange) links.

When you select "DDE (Server)" from the Mode menu, a dialog box will appear prompting for a DDE Application Name and a DDE Topic. In addition to sending serial data, the Wedge can be configured to issue DDE commands to other programs after each input from a serial device is received. DDE commands can be used to force the program that is receiving the data from the Wedge to perform an action like running a macro or updating a chart or graph, etc.. The information that you supply in this dialog box is used to specify the DDE Command Processor for the program that will receive any DDE commands from the Wedge.
Note: This information is needed only if you intend to configure the Wedge to send DDE commands to the application, otherwise it can be cleared out or simply left as is.

See Also: Understanding Dynamic Data Exchange
DDE And The Software Wedge
Specifying DDE Commands (DDE Server Mode)
DDE Examples

The default DDE Application Name is "Excel" and the default DDE Topic is "System". These are the correct parameters for Microsoft Excel's DDE command processor. The DDE Application Name for MS Word is "WinWord" and its DDE Topic is also "System". For MS Access, the Application Name is “MSAccess” and the DDE Topic is the name of your open Access database (without the .MDB filename extension). To find the Application Name and Topic for any other program's DDE command processor, refer to its user documentation.

Note: It is common for applications that support DDE commands to use the application's executable filename (without the extension) as its DDE Application Name and it is also typical to use the DDE Topic "System".



Sending Keystrokes vs. Dynamic Data Exchange

The primary advantage of setting up the Wedge to send keystrokes to another application rather than passing data using DDE is simplicity. By sending keystrokes, no special programming is required in the application that will receive the data. For example, when reading data into a spreadsheet, a common requirement is to have successive readings from a device appear in a column in the spreadsheet. By sending keystrokes you could simply configure the Wedge to send a "Down Arrow" keystroke after each input from your device to move the cursor down one cell and thus be ready for the next input.

See Also: Specifying Pre/Postamble Keystrokes (Send Keystrokes Mode)

A minor disadvantage of sending keystrokes is that the application that is receiving the data must be in the foreground and have the input focus before the Wedge can send any keystrokes to it, thus true background processing is not possible when sending keystrokes.
Sending keystrokes from one application to another is also slightly slower than using DDE.

DDE on the other hand does not require the receiving application to have the input focus therefore all data transfers can occur in the background while you work with other programs in the foreground. One difficulty with DDE is that "linked "data from the Wedge (or any other DDE Server) is always transferred to the same location in the client application (i.e. a "linked" cell in a spreadsheet). To get successive data readings into a column in a spreadsheet you must write a macro in the spreadsheet that runs automatically after each input is received. The purpose of the macro would be to either explicitly request the data from the Wedge or copy each new data item from a *linked* cell and paste it to the bottom of the column in your spreadsheet. One way to accomplish this is to have the Software Wedge issue a DDE Command after each input that causes the spreadsheet to run a macro that either requests the data or copies any linked DDE data into a column.

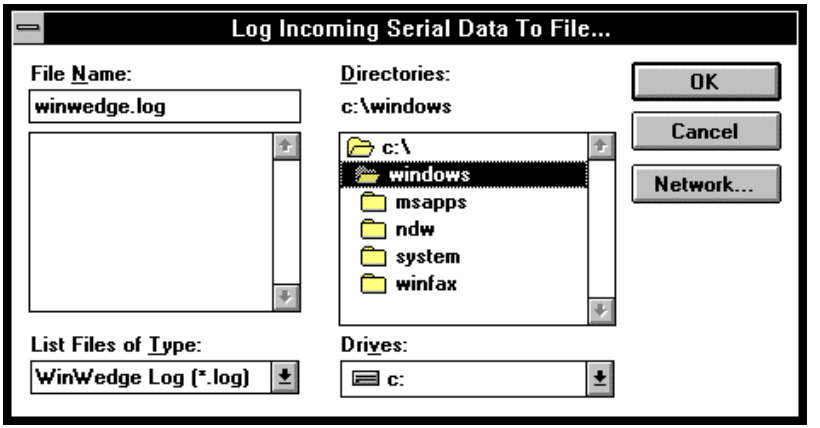
A powerful feature of DDE is that applications that support it can send commands directly to each other and thus you could fully automate a data collection process between your application program and the Software Wedge. This requires a small amount of programming in your application but the extra effort allows you to create extremely sophisticated device interfaces. Also, because all DDE operations can occur in the background, even with a minimized application, DDE operations typically execute much faster than sending keystrokes from one application to another.

Note: **The Software Wedge does <u>not</u> have to be in "DDE Server Mode" in order to accept and process DDE commands.** The Wedge will still execute any DDE commands sent to it even if it is in "Send Keystrokes Mode" or "Log To Disk Mode".

See Also: Understanding Dynamic Data Exchange
DDE Examples
Software Wedge DDE Commands

Log To Disk Mode

When configured in "Log To Disk" mode, all output from the Software Wedge is written to a disk file instead of being sent to another Windows program. If you select "Log To Disk" from the Mode menu, the dialog box shown below will appear where you may specify the disk file that is to receive serial input data. When specifying a file name, you should supply the complete path name including the drive and directory for the file where data is to be stored.

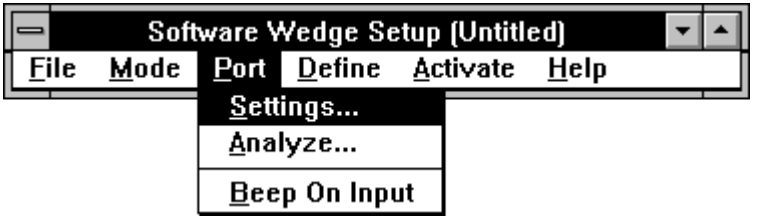


Log To Disk Mode is useful in situations where you would like to capture serial data directly to a disk file for later use instead of transferring the data directly to another application program. After capturing data to a disk file, you could then read the data into another program using any file import facilities that the program offers or by using the File Wedge program that is supplied with the Software Wedge package. Because the Wedge provides a wide array of output data formatting capabilities as well as input data parsing and filtering functions, you could use it to format the serial data so that it will be stored in a manner that is more easily imported into your particular application.

See Also: File Wedge

Note: When the Software Wedge is in “Log to Disk” mode, it can still act as a DDE server. Thus it is possible to capture data to disk ***and*** pass the data to other running applications via DDE simultaneously. The only difference between “Log To Disk” mode and “DDE Server” mode is that the Wedge will not be able to issue DDE commands to another application after each input from a device (i.e. In "Log To Disk" mode you cannot define any Record Preamble or Field Postamble DDE Commands).

The Port Menu



The three PORT menu options and their functions are:

Settings: Displays a dialog box that allows you to select all serial communications parameters, (i.e. serial port, baud rate, parity, number of data bits, flow control, buffer sizes, etc.)
See Also: The Port Settings Dialog Box

Analyze: This option displays a dialog box that allows you to test all serial communications parameters and also view or analyze the structure and contents of data received from your serial input device. The Analyze feature is extremely useful for testing the communications link between your PC and your serial device as well as for determining the structure and contents of all data received from the device. It is highly recommended that you use this feature to test the communications link between the Software Wedge and your serial device before activating the Wedge from the Activate Menu.
See Also: The Port Analyze Menu Option

Beep On Input: This option is a switch that instructs the Software Wedge to beep your PC's speaker when data has been received and is about to be transferred to another application or disk file. Selecting this option toggles this function on or off where "on" is indicated by a check mark to the left of this menu selection.

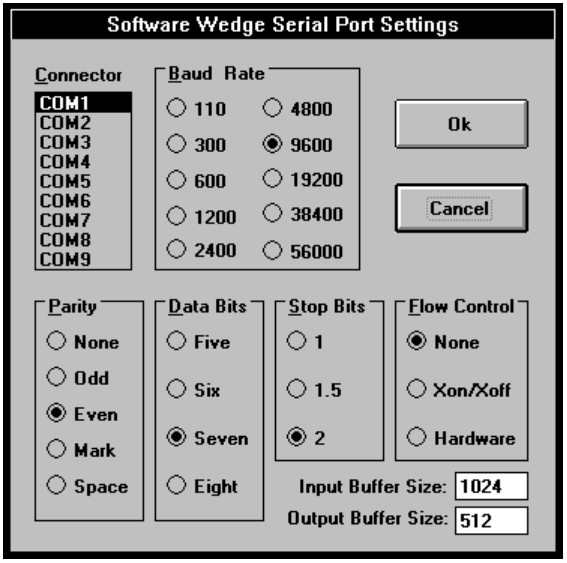
Note: The Software Wedge will also beep whenever a communications error occurs. This includes parity errors, framing errors and buffer overruns. If you do not have the "Beep On Input" option checked and your PC beeps when you receive data through the Wedge, the most likely cause is a mismatch in the communications parameters between the Wedge and your serial device. Because the Wedge treats all characters as valid data, even when a communications error occurs, it is possible to have a mismatch in communications parameters and still receive data that looks correct (although the continual beeping will indicate that something is wrong).

See Also: Diagnosing Serial Communications Problems

The Port Settings Dialog Box

The Port Settings dialog box allows you to select the serial communications parameters required by the device that you are using. All communications parameters selected in the Port Settings dialog box must exactly match the settings for the serial device that you will be using. If you do not know the correct settings for your device, then you should contact its manufacturer for this information. Please do not call for technical support if you do not know the correct parameters for your particular device. For general instructions on how to determine the correct parameters using the Port Analyze feature of the Software Wedge, please refer to the section: Troubleshooting

The Buffer Size options in the Port Settings dialog allow you to specify the size of the receive and transmit buffers. Buffer size values may range between 128 and 32000 bytes. The default input buffer size of 1024 bytes should be sufficient for most simple devices. If you need to input larger amounts of data with no flow control, you may have to increase the input buffer size to avoid losing data. The Output Buffer Size is relatively unimportant and should be left alone for most applications. You may see a slight performance increase in applications that transmit large amounts of data if you enlarge the output buffer size.



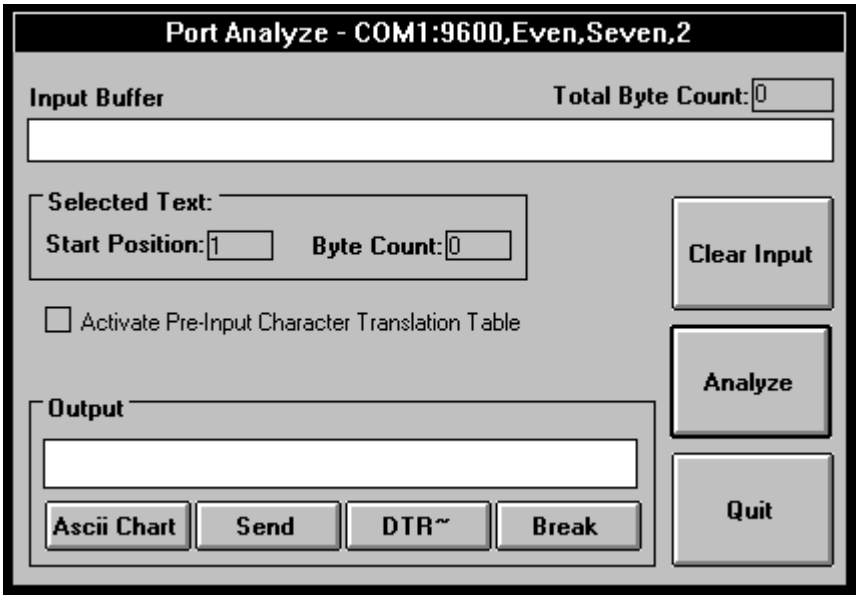
Note: Although the 16 bit version of the Software Wedge supports up to COM9, Windows 3.x only supports COM1 through COM4 using the serial device drivers that come with Windows. To access COM5 and above, you will need special communications device drivers provided by the manufacturer of the serial adapter hardware that you intend to use.

Note: If you need additional serial ports for your PC, TAL Technologies carries a line of inexpensive 2 and 4 port add-on boards. Please visit http://www.taltech.com or call 800-722-6004 for information and pricing. For 8 port boards or PCMCIA serial cards please visit the LINKS page of our web site for links to recommended manufacturers of serial adapter hardware.

The Port Analyze Menu Option

This option presents a dialog box that allows you to test the serial port settings that you chose in the Port Settings dialog box. The Port Analyze dialog box also lets you view or analyze data that is transmitted from your serial device. This feature can be extremely useful when you are configuring the Wedge for your particular application.

If you select this option and the serial port chosen in the Port Settings dialog box exists and is available for use, the following dialog box will appear:



The title bar of the Port Analyze window will show the currently active communications parameters (i.e. serial port, baud rate, parity, databits and stopbits). The "Input Buffer" text box displays all characters received from your serial device. All characters are shown using their displayable ASCII characters including control codes. (i.e. a carriage return (ASCII 13) should appear as a musical note, an ASCII 1 will be a happy face, etc.) An ASCII chart is available to help decipher the meanings of characters in the input buffer and also to select characters to be placed in the "Output" text box. To display the ASCII chart, click the button marked "ASCII Chart".

Directly above and to the right of the Input Buffer is a text box labeled "Total Byte Count" that will contain a count of all characters currently in the input buffer.

The "Clear Input" button is used to clear the input buffer and reset the total byte count to zero. Note: The Input Buffer text box can display a maximum of 32767 characters at a time.

An automatic analyze feature allows you to have the Software Wedge make an educated guess at the structure of your input data records. The Software Wedge can be configured to parse and filter your serial data, however in order to have it do this, you will first need to know what the *record structure* is for your data. If you press the "Analyze" button while there is data in the input buffer, the Software Wedge will offer a guess at the record structure and ask you if you would like to automatically pre-configure the wedge using the guessed at record structure. Note: The capability of the auto analyze feature is limited to recognizing only the most common record structures and thus you may not always be able to use it to pre-configure the Wedge. You are entirely free to define your input record structure in any way that you like.

See Also: Defining the Input Data Record Structure
Understanding The Data From Your Serial Device

When viewing data in the input buffer, you can select portions of the data using your mouse or the cursor control keys. When you do this, the starting byte position of the selected data in the input buffer is displayed next to the label "Start Position" and the byte count for the selected data is displayed next to the label "Byte Count" (located inside the frame labeled "Selected Text"). This feature allows you to measure the lengths of specific data fields in your input data records.

To copy data from the Input buffer to the Windows clipboard, select the data that you want to copy then hold the control key (Ctrl) down and press the Insert key.

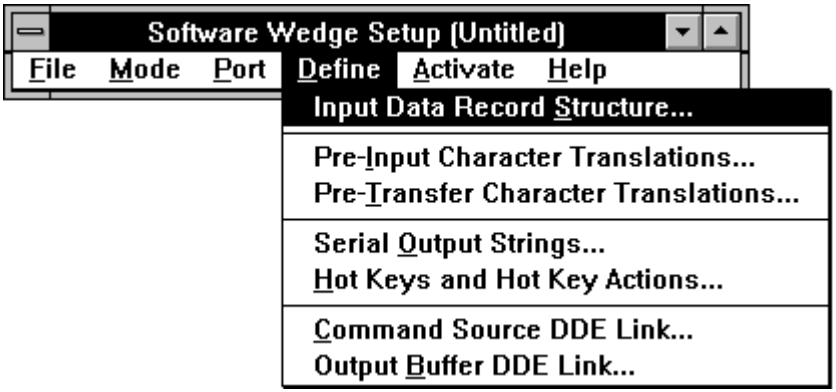
In the center of the Analyze window is a check box with the caption "Activate Pre-Input Character Translation Table". When this option is checked, all data received will be translated using the "Pre-Input Character Translation Table" before being displayed in the input buffer. If it is not checked, the data that appears will be the exact input from your serial device with no translations.

See Also: Translation Tables

Inside the frame with the caption "Output" is a text box and four buttons labeled "ASCII Chart", "Send", "DTR~" and “Break”. The text box is an output buffer where you can enter text to be sent out to your serial device. Many devices allow or require you to send them command strings or control codes, for example to request data. You may type and edit data directly in the Output text box. You may also select specific characters or control codes to be placed in the output text box from an ASCII chart that is available by pressing the "ASCII Chart" button. To transmit the data in the output text box, click your mouse on the "Send" button. To clear the output buffer, select all text in the Output text box with your mouse or cursor control keys and press the Delete key.

The button marked "DTR~" is called the *Data Terminal Ready Line* Toggle button. When this button is pressed, it turns off the serial port DTR line for approx. 100 ms. The button marked "Break" can be used to issue a serial Break signal. These buttons are provided because some types of serial devices interpret the toggling of the DTR line or the issuing of a Break signal as a request for data.

The Define Menu


Software Wedge Setup (Untitled)
File Mode Port Define Activate Help
Input Data Record Structure...
Pre-Input Character Translations...
Pre-Transfer Character Translations...
Serial Output Strings...
Hot Keys and Hot Key Actions...
Command Source DDE Link...
Output Buffer DDE Link...

The seven options in the DEFINE menu and their functions are:

Input Data Record Structure

This option presents a series of dialog boxes that allow you to define the structure of input data from your serial device. The dialogs presented also allow you to specify how you would like the Wedge to parse, filter and format incoming serial data before transferring it to another application or disk file. When you define the input data record structure, you can also add additional data or keystrokes to each input field or specify DDE commands to be issued following the input of each field.

See Also: Defining The Input Data Record Structure

Pre-Input Character Translations

Displays a translation table that allows you to translate incoming ASCII characters to other characters before they are acted upon by the Wedge. This is an extremely powerful concept that allows a great deal of flexibility in the way incoming data is parsed or filtered by the Wedge.

See Also: Translation Tables

Pre-Transfer Character Translation Table

Displays a translation table for translating characters to keystrokes when the Software Wedge is in *Send Keystrokes* mode or for translating characters to other characters when the Software Wedge is in either *DDE Server* mode or *Log To Disk* mode. Pre-Transfer translations are performed after the Software Wedge has parsed, filtered and formatted an incoming data record, just before the data is transferred to another application or disk file.

See Also: Translation Tables

Serial Output Strings

Opens a window where you can pre-define character strings to be transmitted back to your serial device by the Software Wedge. Several output strings may be defined including an *acknowledgment* string as well as up to 20 "button controlled" output strings and one "timer controlled" output string.

See Also: Defining Serial Output Strings

Hot Keys and Hot Key Actions

This option presents a dialog box where you can define up to 50 different hot keys that can be used to perform any of thirteen different actions including transmitting strings out the serial port, toggling the state of the RTS or DTR serial I/O lines and enabling, disabling and resetting the Wedge.

See Also: Defining Hot Keys and Hot Key Actions

Command Source DDE Link

This option allows you to specify a DDE Server Item in another application program that will be used to supply commands to the Software Wedge. For example, you could specify DDE parameters that reference a cell in a spreadsheet program so that whenever a valid Software Wedge DDE command is entered in the specified spreadsheet cell, the Wedge would execute the command.
Note: It is extremely rare that you will ever need to use this option because almost all applications that support DDE provide a much better method for sending DDE commands to another program. This option is available mainly to provide an alternative way to send a command to the Wedge from applications that have no other mechanism for doing so.

See Also: Defining A Command Source DDE Link
Software Wedge DDE Commands

Output Buffer DDE Link

This option allows you to specify a DDE Server Item in another application program that will be used to supply serial output data to the Software Wedge. For example, you could specify parameters that reference a cell in a spreadsheet program so that whenever any data is entered in the specified spreadsheet cell, the Software Wedge would transmit the data out the serial port.
Note: Similar to the "Command Source DDE Link" above, it is extremely rare that you will ever need to use this option. In most cases where you need to transmit data out the serial port from an application, you would send DDE commands to the Wedge instead of using this option. Again this feature is provided mainly for compatibility with the few applications that support DDE but cannot send DDE commands directly to another program.

See Also: Defining An Output Buffer DDE Link

Defining the Input Data Record Structure

Most serial devices transmit data in some pre-defined *record structure* with each record containing one or more parts or *fields* of data. Because the Software Wedge has no way to know how the data from your particular device is structured or how to determine what parts of the data are important to your application and which parts should be ignored, you must first define these parameters before activating the Wedge. The *Input Data Record Structure* option in the "DEFINE" menu allows you to first define the record structure of your data and then define how the Wedge should parse, filter and format the individual data fields within each record before sending the data on to another application. This option also provides a mechanism for adding additional data, keystrokes or DDE commands to the serial data before or after each data field is sent from the Wedge to your target application program.

When defining the input data record structure, several basic descriptors must be specified starting with the events that determine the start and end of each data record. When you select *Input Data Record Structure* from the Define menu, the following dialog box appears:

The Start Of Record Event

The *Start Of Record Event* options specify the event that triggers the Wedge to start accepting characters as part of each new data record (for the first data record and also after the end of each record). If you select *Any Character Received* as the start of record event, the first character received will be considered to be the start of each record. All subsequent characters received will be treated as part of the record until an *End Of Record Event* occurs. The *Alpha/Numeric Char Received* option instructs the Wedge to ignore input until a valid Alpha/Numeric character is received (i.e. non control characters or chars with ASCII codes greater than 31 and less than 127). The *Numeric Char Received* option causes the Wedge to wait until a numeric character is received (i.e. 0123456789.-).

The *Special Char Received* option allows you to specify which characters are to signal the start of each new data record. If you use this option, you may enter one or more characters in the text box next to this option. To enter control codes in the text box, press the *ASCII Chart* button and choose the specific character(s) from the displayed ASCII Chart. If you enter more than one character, the reception of any of the characters will trigger the start of a record. For example, if you specify "AB", either an "A" or a "B" will signal the start of a record (not the complete string "AB").

See Also: Pre-Input Character Translation Table

The End Of Record Event

Similar to the *Start Of Record Event*, the *End Of Record Event* options allow you to specify the event that will signal the end of each input data record. The Wedge will not transfer any data to a target application or disk file until the selected *End Of Record Event* occurs. Therefore you should select the *End of Record Event* that will always reliably determine when each data record ends.

If each data record is terminated by a carriage return or carriage return linefeed, the *Carriage Return or CrLf Received* option should be chosen. Carriage returns and line feeds are **not** removed automatically if this option is chosen. If you do not want these characters as part of your data, use the "Pre-Transfer Character Translation Table" to remove them by translating them to "Nul".

See Also: Translation Tables

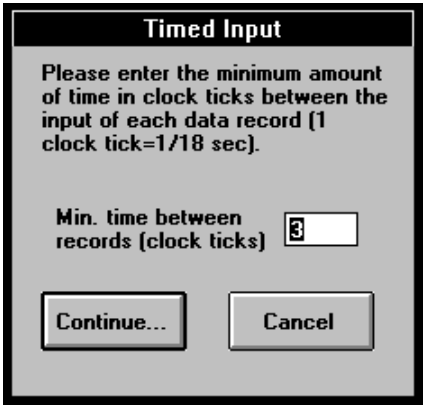
If the data records transmitted by your device always consist of a fixed number of bytes with one or more fixed length data fields, then the option *Fixed Number of Bytes Received* should be chosen. If you choose this option, you will be prompted with the following dialog box to enter the complete record length for your data records.

See Also: Pre-Input Character Translation Table
The Port Analyze Menu Option

If your incoming serial data is not always transmitted as fixed length strings and there also is no specific character within the data that can be used to signal the end of each data record but the data is transmitted in bursts such that there will always be a small time delay between data inputs, then *Time Delay Between Records* should be chosen.
Note: Many types of bar code scanners transmit data in this manner.

If you select this option, you will be prompted to enter the minimum amount of time (in 1/18 second increments) between the arrival of each new data record. You should specify the smallest time delay value possible to accurately determine the end of an input data record. If you specify too large a value, then you risk the possibility of two or more inputs being interpreted as one long input. The default value of 3/18 of a second is almost always the best value to use for the types of devices that the Wedge is typically used with.

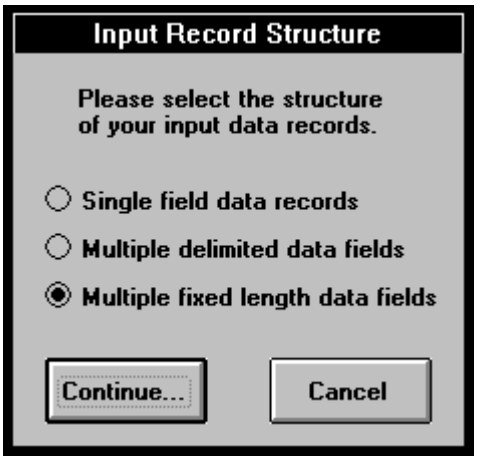
Note: If the actual time delay between data inputs from your serial device is less than 1/4 second, this option may not be completely reliable. If another application is busy performing some other task that takes longer than the real delay between incoming data records, the Wedge will not always be able to pre-empt the other task and measure time with complete accuracy.


Timed Input
Please enter the minimum amount of time in clock ticks between the input of each data record (1 clock tick=1/18 sec).
Min. time between records (clock ticks)
3
Continue...
Cancel

The *Special Char Received* End of Record Event option allows you to specify characters that will signal the end of an input record. If you choose this option, you may enter one or more characters in the text box next to this option. To enter control codes in this text box, press the "ASCII Chart" button and choose the characters from a displayed ASCII Chart. The reception of any of the characters entered will signal the end of an input data record.
NOTE: Special characters are **not** automatically removed if this option is chosen. If you do not want these characters to appear as part of your data, then you can use the "Pre-Transfer Character Translation Table" to remove them by translating them to "Nul".

See Also: Pre-Transfer Character Translation Table

If you choose either *Carriage Return or CrLf Received*, *Time Delay Between Records* or *Special Char Received* as the End of Record Event, you will also be prompted to select a "record structure" for your data from the following dialog box:



The three possible record structures and their meanings are listed below:

Single field data records

Single field data records means that each entire data record should be considered as a single entity thus no parsing should be performed on the data.
Note: The maximum length for a single data field is 32766 bytes.

NOTE: Some types of serial devices produce an output that may be too complex or too inconsistent to be parsed into a set number of "fields" by the Software Wedge. For these types of devices, it may be easier to define the entire output from the device as a single data field and then use the capabilities of the target application (the macro language in Excel for example) to parse the data received from the Software Wedge.

See Also: Understanding the Data From Your Serial Device
Pre-Input Character Translation Table
Sending Keystrokes Vs. Dynamic Data Exchange

Multiple delimited data fields

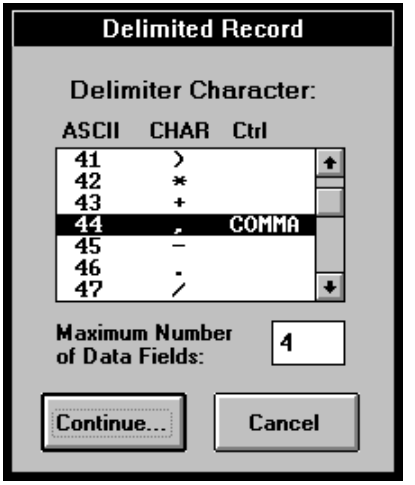
This record structure option specifies that your data records consist of two or more unique *fields* that should be parsed based on the position(s) of a delimiter character within each record. For example the following record consists of four data fields delimited by commas and with a carriage return linefeed at the end of the record:
12345, 9090, Value=123, 98765<CrLf>

If you select this option, you will be prompted with the dialog box shown below to specify the delimiter character used to separate data fields and also enter the maximum number of data fields that could be contained in a single data record.

The Software Wedge supports records with up to 40 data fields. If more than the specified number of fields are received for a particular record, the additional data is discarded. If less than the specified number of fields are received, the remaining fields will be nul.

Note: If you are trying to interface a device that outputs data records with more than 40 data fields per record, there are several methods that you can use to overcome the 40 field limit in the Wedge. When the Wedge is in "Send Keystrokes" mode, one method is to define the record structure as "Single Field" data records and then using the "Pre-Transfer Translation Table" to translate the delimiter characters to cursor navigation keystrokes such as the Right Arrow, Down Arrow, Tab or Enter keys. Another method involves passing each data record as a single data field (in either "Send Keystrokes" or "DDE Server modes") and then using the macro language in the target application to parse the data and put the individual fields wherever you would like them to go.

See Also: Defining The Input Data Record Structure
Overcoming the 40 field limit in the Wedge.



Special Considerations Regarding Delimiter Characters

All delimiter characters will be removed from the input data and thus will not appear as part of each data field. If you choose a space character as your delimiter, the Software Wedge will treat strings of consecutive spaces as a single delimiter. Also, if a chosen delimiter character matches a character that signals the end of an input record, (for example, if you choose a carriage return as your delimiter character and you also chose "Carriage Return or CrLf Received" as your End of Record Event), the Software Wedge will treat the character as the delimiter until the specified number of data fields have been received after which the next delimiter/end of record character will signal the end of the record.
See Also: Defining The Input Data Record Structure

For example, suppose you have a device that transmits two lines of data with a carriage return at the end of each line. If you specify the “End of Record Event” as “Carriage Return or CrLf Received” and also specify a carriage return as your delimiter and then specify “2” for the “Maximum Number of Data Fields”, the Wedge will treat the data from the first line as the first data field and the data from the second line as the second data field.

See Also: Understanding the Data From Your Serial Device

In cases where your input data records contain two or more different delimiter characters, you can use the "Pre-Input Character Translation Table" to convert all delimiters to a single character. For example, consider the following record:

Instrument# 123, Reading#3=23.45<CrLf>

Normally this data record would be interpreted to contain two fields: "*Instrument# 123*" and "*Reading#3=23.45*", with a single comma delimiter between the two. If we were to use the Pre-Input Character Translation Table to convert both pound signs (#) and equal signs (=) to commas, then after the translation, the record would appear to the Software Wedge as:

Instrument, 123, Reading,3,23.45<CrLf>

The translated record thus now contains five individual, comma delimited, data fields.

See Also: Pre-Input Character Translation Table
The Port Analyze Menu Option

Multiple fixed length data fields

Multiple fixed length data fields should be chosen as the record structure if you need to parse your data records on the length of each data field. If you use this option, you will later be able to specify the exact length (number of characters) contained in each data field. This option should be used only in cases where you can always rely on each data field to contain a fixed number of bytes. This option is the default if you chose "Fixed Number of Bytes Received" as the End of Record Event.

See Also: Pre-Input Character Translation Table

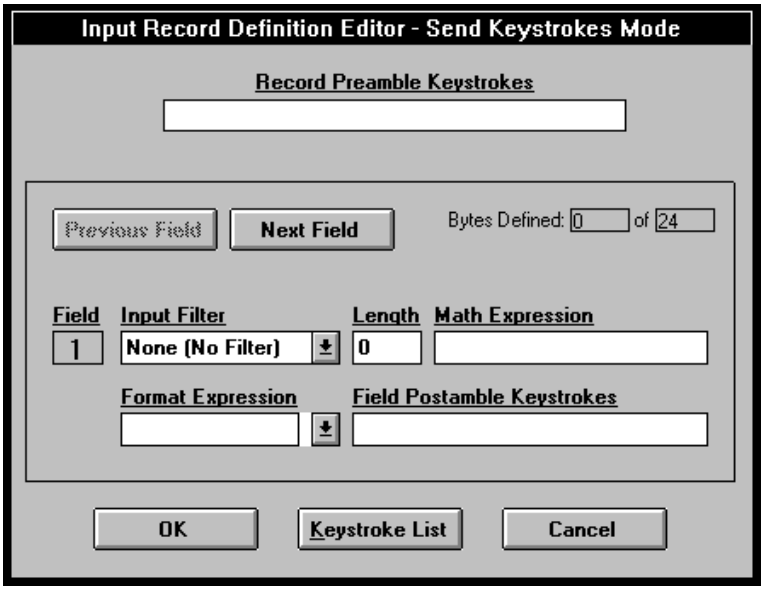
Specifying Filters, Output Formats, and Field Lengths

After you specify the Start and End of Record Events and choose the basic structure of your data records, an "Input Record Definition Editor" dialog box will appear. This dialog box is where you specify a filter to be applied to each field, the input length of each field (for fixed length fields) an optional math expression to be applied to the data and an optional output data format expression for each field.

This dialog box also allows you to specify a record *Preamble* and field *Postambles* that perform different functions depending on the mode that you chose for transferring data.
In *Send Keystrokes* mode, the record preamble and field postambles are additional keystrokes that are issued before or after each data field is sent to another program. In *DDE Server* mode, the preamble and postambles are DDE commands that are issued to the application that you specified as the *DDE Command Destination Application* when you chose DDE Server from the Mode menu. In *Log To Disk* mode, the preamble and postambles are additional characters or control codes that you want written to disk along with the serial input data.

See Also: Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode)
DDE Server Mode
Specifying DDE Commands (DDE Server Mode)
Specifying Pre / Postambles (Log To Disk Mode)

Note: Some of the controls in the Input Record Definition Editor may not be displayed or may have different captions than the example below depending on the chosen record structure and also depending on the selected data transfer mode.



The lower half of the dialog box is where you specify an "Input Filter" to be applied to each data field in your input records as well as an optional “Math Expression”, an optional "Format Expression" and a "Field Postamble". In cases where each data field consists of a fixed number of bytes, you must also specify a field "Length".

Under the label "Field" is a status box indicating the number of the current data field that you are defining a Filter, a Length, a Math Expression, a Format Expression, and a Field Postamble for. If your data records consist of more than one data field, there will be two buttons marked "Next Field" and "Previous Field" in the dialog box. These buttons are used to scroll forward and backward through the parameters for each data field, (the first field is Field(1); pressing the "Next Field" button will display parameters for Field(2), etc.).

Selecting Filters

There are six choices for the "Filter" that can be applied to each data field; None, Ignore This Field, Numeric Data Only, Hex To Decimal, Oct To Decimal and Binary To Hex. Specifying "None" means that no filter is to be applied to the data for the current field, thus all characters are to be accepted as received. If you select "Ignore This Field", then no data is accepted and the entire field is ignored. The Ignore filter is thus used to ignore or remove data fields that are not required by your application. Specifying "Numeric Data Only" causes all Alpha characters and control codes to be filtered out and only allows characters: **0123456789-.** to be accepted as valid data.

The “Numeric Data Only” filter is especially useful when reading numbers into a spreadsheet. Leading spaces or non-numeric characters can cause data to be interpreted as a label instead of a number. The Numeric filter can also be used to reduce the number of data fields that need to be parsed as well as remove trailing carriage returns and linefeeds. For example, suppose you had a device that transmits the following comma delimited, carriage return terminated data records:
X=0123.45,Y=0321.22<cr>
If you are only interested in the two numeric values, you could define the record structure as "multiple delimited data fields" choosing the comma as the delimiter and specifying two as the maximum number of fields and finally applying a "Numeric Filter" to each of the two data fields. In this case you would end up with the two numeric values only and the additional characters "X=", "Y=" and the carriage return (<cr>) would be removed by the numeric filter.

The “Hex To Decimal" filter performs a Hexadecimal to Decimal conversion on the data contained in the field. For example, the string "FF" would be converted to "255".
The "Oct To Decimal" filter likewise performs an Octal to Decimal conversion on the data.

The “Binary To Hex” filter converts each byte of data to a two byte hexadecimal string. For example, an ASCII character 255 would be converted to "FF" and an ASCII 0 character would be converted to "00". The “Binary To Hex” filter allows the Wedge to work with raw binary data and not just ASCII text data. Note: You can use the “Binary To Hex” filter in conjunction with the "&h( )" math function to easily convert binary data to decimal numbers.

See Also: Mathematical Functions and Operators

Specifying Math Expressions

The Math Expression text box allows you to specify a mathematical expression that can be applied to numerical, and in some cases, string data received from a serial device. Suppose for example that you were taking readings from a measuring instrument and you needed to multiply each reading by 100 before transferring it to another application program. You could accomplish this by entering the formula: **FIELD(1)*100** as the math expression for field(1).

The Software Wedge supports all standard math functions including addition, subtraction, multiplication, division, integer division, modulo arithmetic, trigonometric functions, logical functions, absolute values, exponential and logarithmic functions, square root and logical bitwise operations. Several string functions are also supported including upper and lower case conversion functions.

A "hex to decimal" function "&h()" also allows you to convert hexadecimal data to decimal numbers. The &h() function can be used in conjunction with the "Binary To Hex" filter to convert pure binary data into decimal numbers. For example, suppose you have a device that transmits a string that contains a two byte binary value that would appear in Field(1) in the Wedge. If you apply a "Binary To Hex" filter to Field(1) and then use the expression "&h(Field(1)" as the math expression for field(1), the Wedge would convert the two byte binary value into a decimal number.

See Also: Mathematical Functions and Operators

Defining Output Formats

Similar to the way most spreadsheets allow you to format numbers and text, each data field defined in the Wedge can be formatted before being outputted to another application or disk file. To define a format for a particular data field, you may use either one of eight pre-defined numeric formats or you may custom specify a format by entering a string of special formatting characters in the text box marked "Format". It is possible to specify multiple format expressions for each field such that different formats are used depending on the contents of the field.

Note: You do not have to specify a format expression. If the format expression is left blank, no formatting is performed on the data.

See Also: Format Expressions

Specifying Field Lengths

The text box labeled "Length" is where you specify the exact number of data bytes that will be received for a particular field. The "Length" text box will only appear if you selected either "Fixed Number of Bytes Received" as the "End Of Record Event" or "Multiple Fixed Length Data Fields" as the record structure. The field length thus specifies the exact number of bytes in each specific data field in the input record. Field lengths may range from 1 to 32766 bytes. For fixed length records, the sum total of all field lengths entered should equal the total length of each input record.

See Also: Pre-Input Character Translation Table

In the middle toward the right side of the Input Record Definition Editor (labeled "Bytes Defined") are two status boxes that display the total of all specified field lengths as well as the total record length of your data records (specified earlier for fixed length records). This information is only visible when the end of record event is a fixed number of bytes received.

Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode)

When the Wedge is in Send Keystrokes Mode, the input Record Definition Editor will have a text box marked "Record Preamble Keystrokes" in the upper half of the dialog window. Also, for each data field that you have defined, you may enter a sequence of "Field Postamble Keystrokes". Preamble and Postamble keystrokes are additional keystrokes that you would like issued before or after each data field is sent to another Windows application. They are typically used to control the receiving application so that the serial data ends up where it is supposed to go.

The "Record Preamble Keystrokes" are keystrokes that you would like issued immediately before the first data field from your serial device is transferred to your application program. For example, the preamble keystrokes could be used to move the cursor to a specific location in the target application just before any data is transferred to it.

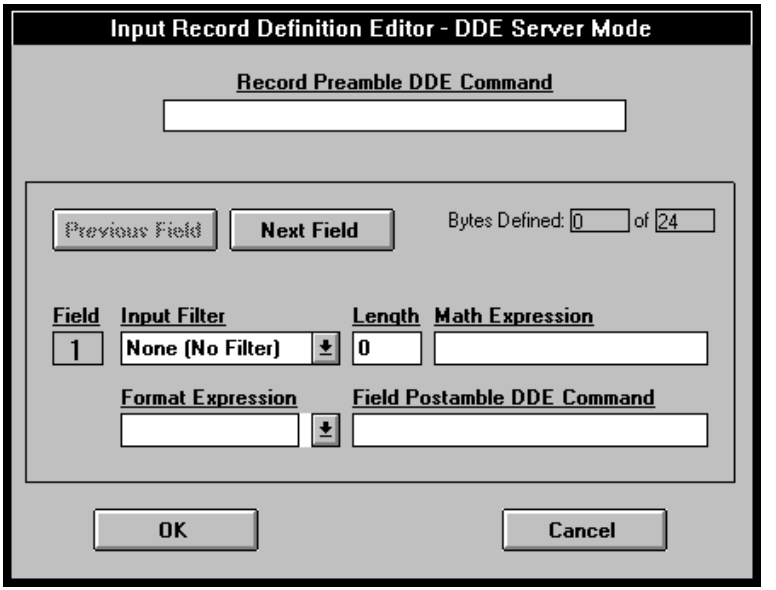
The "Field Postamble Keystrokes" are issued immediately following the data for each particular data field. Field Postamble keystrokes are usually used to navigate around in the application that is receiving the data. For example, if you are reading data from a device into a spreadsheet and you want successive readings entered in a single column, you could issue a field postamble keystroke consisting of a {DOWN} arrow. This would cause the cursor to move to the next cell down in the spreadsheet after each input from the device. You define Record Preamble and Field Postamble keystrokes to mimic the keys you would press if you were typing the data manually on the keyboard

When specifying Record Preamble and Field Postamble keystrokes, you must follow the rules outlined in the section: Keystroke Macro Rules
The button marked *Keystroke List* allows you to select individual keystrokes from a list while editing keystroke macros. All keystrokes in the list conform to the proper Keystroke Macro Rules. Preamble and Postamble Keystroke macros may also contain special date and/or time stamps.

See Also: Date and Time Stamps In Macros

Specifying DDE commands (DDE Server Mode)

When using the Software Wedge in DDE server mode, the Input Record Definition Editor will appear as it does below.



All of the controls in the above dialog box are the same as when the Wedge is in "Send Keystrokes Mode" except that instead of supporting *Record Preamble Keystrokes* and *Field Postamble Keystrokes*, the dialog box allows you to enter a *Record Preamble DDE Command* and *Field Postamble DDE Commands*. Most applications that support DDE also support a set of *DDE commands* that may be issued to them by other application programs (such as the Wedge). DDE Commands allow other programs to either control the application or force it to perform some type of action like running a macro or subroutine. DDE commands will be unique to a particular application and should be thoroughly documented in the application's users manual.

Excel for example recognizes a wide range of DDE commands that allow other Windows applications to take complete control of it. In fact, Excel even recognizes every one of the function names listed in the Excel Function Reference as valid DDE commands. For example, Excel supports a function called "RUN(reference)" that can be used to run any macro or subroutine defined in an Excel macro or module sheet where "reference" is the name of the macro or subroutine in quotes. If you create an Excel subroutine named: "DoSomething" and then enter the command: [RUN("DoSomething")] as a Field Postamble DDE command in the Software Wedge, Excel will execute the "DoSomething" subroutine immediately after data for the field is updated by the Software Wedge.

This concept is extremely powerful because it allows you to use the entire Excel macro language including Visual Basic for Applications to perform nearly any operation on data received from the Software Wedge. This includes chart or graphing functions, calculations based on received data, or logical decision making - all in real time as data is being received through the serial port.

Any "Record Preamble DDE Command" specified will be issued immediately before any of the input data fields are updated in the Wedge and each "Field Postamble DDE Command" is issued immediately after the data for the particular field is updated in the Wedge.

In almost all cases you would probably only want to issue a single Field Postamble DDE Command after the very last data field that you have defined in the Wedge. The purpose of the command would typically be to force another application to run a macro or subroutine. The macro or subroutine in the other program would then request data back from the Wedge (i.e. the input data fields containing the serial data) and then go on to process the data.

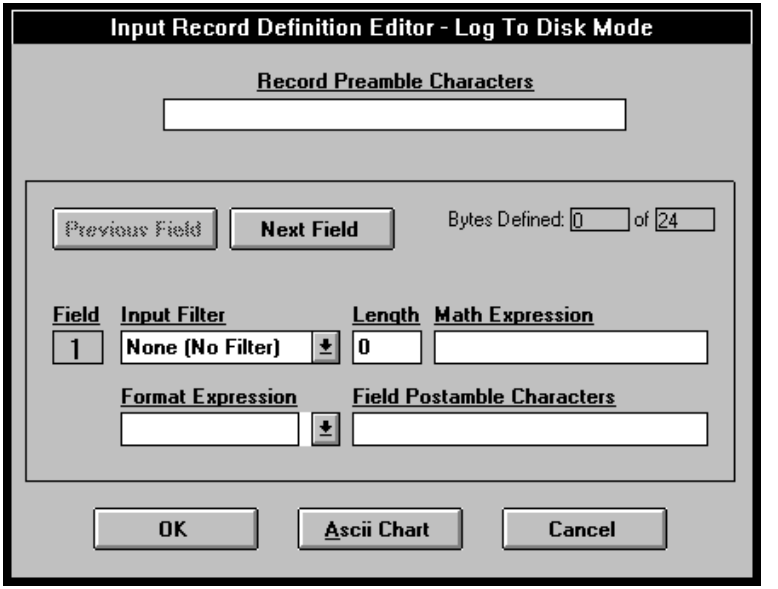
To learn what commands a particular application supports, you will have to refer to its user's manual. You can also refer to the DDE Examples section of this manual for specific examples for applications like Excel, Access, Visual Basic, FoxPro, Quattro Pro, Lotus 123 and Microsoft Word.

Note: Some applications that support DDE provide a way to program the application to "sense" when new data is received from a server in a DDE conversation. Excel for example supports two function “SetLinkOnData()” and "OnData()" that can be used to cause Excel to automatically run a specific macro or subroutine whenever new data is received from a DDE server. This type of capability allows you to do the same thing that a Field Postamble DDE Command would do without requiring the Wedge to actually issue a Field Postamble DDE Command. In other words instead of the Wedge sending a command to force a macro to be run, the other application would be set up to listen for changes in DDE data and when the data changes it would automatically run a macro.

See Also: DDE Examples
Understanding Dynamic Data Exchange

Specifying Pre / Postambles (Log To Disk Mode)

When using the Software Wedge in "Log To Disk Mode", the Input Record Definition Editor dialog box will appear as below.



Again, most of the controls in this dialog box are the same as for either "Send Keystrokes" or "DDE Server" modes except that in Log To Disk Mode, the dialog allows you to enter a "Record Preamble Characters" and "Field Postamble Characters". Because the Software Wedge will be writing all output to a disk file instead of to another application program, the Record Preamble and Field Postambles may only consist of additional ASCII characters that are to be written to disk along with the serial data. For example, you may wish to write a carriage return - linefeed pair (ASCII 13 & ASCII 10) after each individual data field as they are being saved to disk. To specify preamble or postamble characters that cannot be typed on your keypad, an ASCII chart is available by clicking on the button marked "ASCII Chart".

Note: Preambles and Postambles in *Log To Disk* Mode may also contain special date and/or time stamps.

See Also: Date and Time Stamps In Macros

Translation Tables

The Software Wedge provides two translation tables; a *Pre-Input Character Translation Table* and a *Pre Transfer Character Translation Table*. The two translation tables allow you to translate characters received from your serial device at two different points, before the data is acted upon by the Wedge, and just before data is transferred to another application program or disk file - after the data has been parsed, filtered and formatted.

Pre-Input Character Translation Table

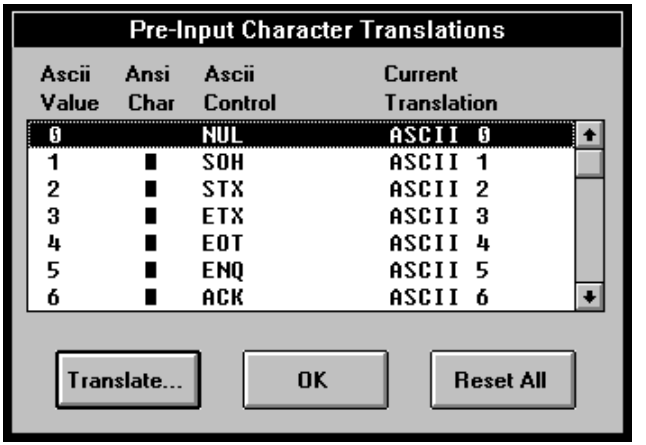
The Pre-Input Character Translation Table allows you to translate incoming ASCII characters to other ASCII characters before they are acted upon by the Wedge. This is an extremely powerful concept because it can dramatically alter the operation of the Wedge as well as the way in which you think about your serial data when you configure the Wedge.

For example, suppose that your serial device transmitted data records containing multiple delimited data fields but some of the fields were delimited with commas and others were delimited with semi-colons. In this case you could use the Pre-Input Character Translation Table to translate semi-colons to commas and then define the "Record Structure" in the Wedge as "Multiple Delimited Data Fields" choosing the comma as the delimiter. In this case the Wedge would act on all semi-colons as if they were comma delimiters.

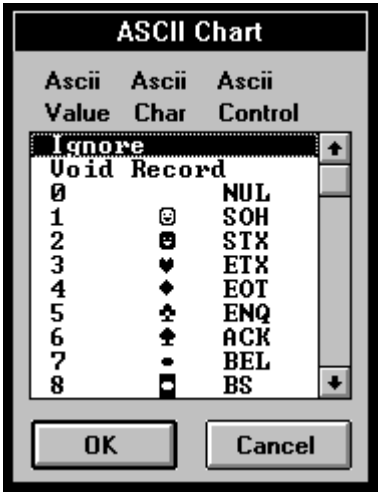
See Also: Special Considerations Regarding Delimiter Characters

You can also use the Pre-Input Translation Table to strip out specific characters from the serial data or even remove an entire data record if it contains a specific character.

If you select *Pre-Input Character Translation Table* from the *Define* menu, the following dialog box containing the translation table will appear:



The translation table shows the ASCII values for all possible characters as well as their corresponding ANSI characters, ASCII control code mnemonics and the current translation for each character. To translate characters, select the character that you need to translate with your mouse and click the "Translate" button. This will cause the following ASCII Chart to be displayed where you can select a specific translation. The "Reset All" button in the translation table causes all translations to revert to their default settings.



The first two choices "Ignore" & "Void Record" in the ASCII Chart do not represent actual characters but instead have special meanings when selected.

"Ignore" causes the translated character to be ignored by the Software Wedge. When a character translated to "Ignore" is received, it will be discarded and thus it will not appear in the input data. Also, because the character is being ignored, its presence is not counted when reading data into a fixed length record or a fixed length data field. Thus, “Ignore” removes the character from the input data before it reaches the Software Wedge.

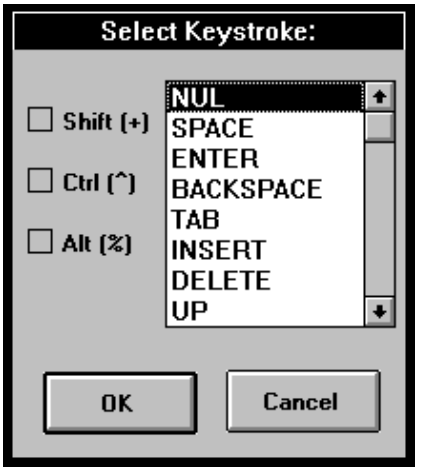
"Void Record" causes the Wedge to invalidate the entire current record. If a character that has been translated to "Void Record" is present anywhere in an input data record, the entire record is completely ignored and will not be transferred to another application program or disk file. This feature can be useful in situations where you would like to reject certain data records from a device. For example, suppose you had a device that always transmitted a special "initialization record" each time the device is turned on. If the initialization record contained any character that would never appear in any following record, you could use the "Void Record" translation to remove the unwanted initialization record.

Pre-Transfer Character Translation Table

The Pre-Transfer Character Translation Table is used to translate characters after the Software Wedge has parsed, filtered and formatted your data; i.e. just before the data is sent to another application program or disk file.

The “Pre-Transfer Translation Table” is almost identical to the "Pre-Input Character Translation Table" except that this translation table will work differently depending on the current output mode that the Software Wedge is set for i.e. "Send Keystrokes", "DDE Server" or "Log To Disk” Mode.

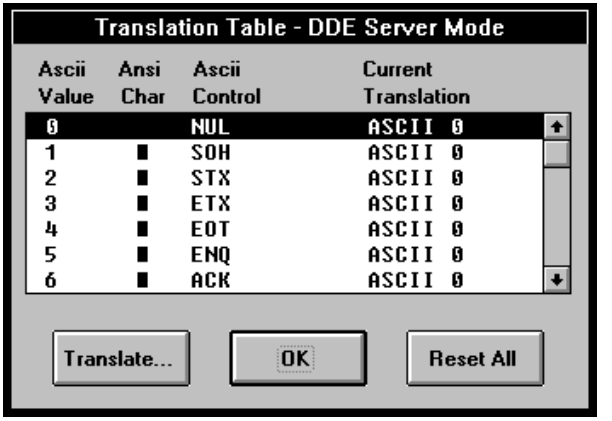
If the Wedge is in "Send Keystrokes Mode", then this translation table is used to convert individual characters to specific keystrokes. Serial data consists of ASCII characters. Because many ASCII characters do not correspond to a specific keystroke, the Pre-Transfer Character Translation Table provides a way to map characters to specific keystrokes. To translate characters, simply select the character that you need to translate with your mouse and then click the "Translate" button. This will display the following "Keystroke Selection" dialog box that allows you to select a specific keystroke or key combination.



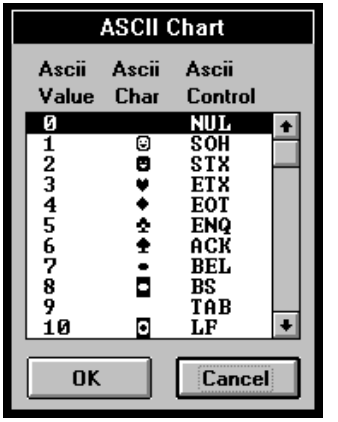
The keystroke selection dialog box contains a list of all keystrokes as well as three check boxes that allow you to indicate the toggle key states to be used in conjunction with the selected keystroke.

The first item in the keystroke list, "NUL", does not represent an actual keystroke but instead is used to remove unwanted characters such as carriage returns and control codes or other characters before data is sent to another application program.

If the Software Wedge is in either "DDE Server Mode" or "Log To Disk Mode" then the Pre-Transfer Character Translation Table will appear as shown below. This translation table allows you to translate characters to other ASCII characters.



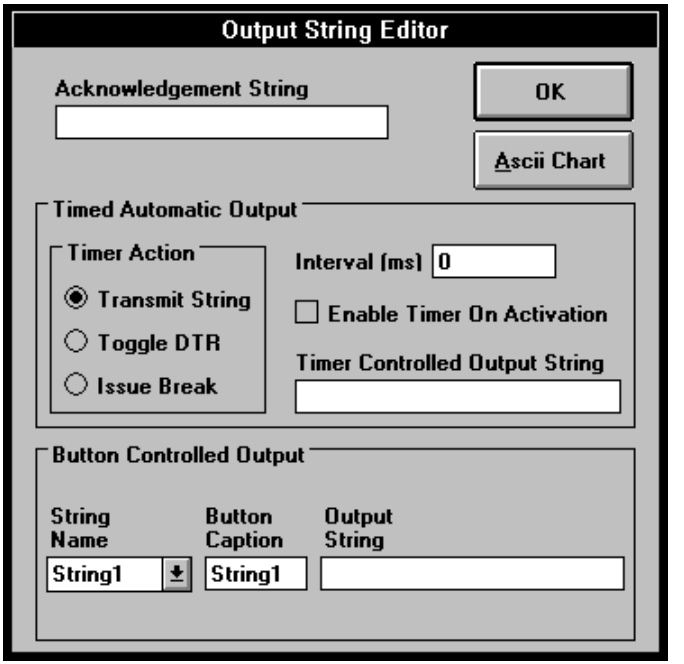
When you click on the "Translate" button to translate a character, the following ASCII Chart will appear allowing you to translate the selected character to another ASCII character. This ASCII Chart is similar to the one displayed for the Pre-Input Translation Table except that it does not have an "Ignore" or "Void Record" entry.



NOTE: When using the Wedge in DDE Server Mode, characters translated to "Nul" in the Pre-Transfer Character Translation Table will be removed from the outputted data. Translating characters to NUL can thus be used to remove unwanted characters. In Log To Disk Mode, all Nul characters (ASCII 0) and characters translated to "Nul" are written to disk along with the rest of your data.

Defining Serial Output Strings

Selecting "Output Strings" from the Define menu opens the dialog box shown below where you may define an acknowledgment string, a timer controlled output string and up to 20 button controlled output strings that can be sent to your serial device while the Software Wedge is active.



The Acknowledgment String is a character string that is automatically sent out the serial port after each complete data record is received from your serial device. The capability to send an Acknowledgment string was originally intended for those devices that require an acknowledgment (an ACK character for example) but it could also be used as a way to continually request data from a device that can be polled by sending it a character string.

A Timer Controlled Output String may also be defined that is automatically transmitted at regular intervals. The timer interval value may range from 50 to 99,999,999 milliseconds (i.e. 1/20th of a second to once every 27 hours). A check box also allows you to specify if timed automatic outputs are initially enabled as soon as you activate the Wedge. If a timer controlled string is defined, a menu item in the Software Wedge window (displayed after the wedge is activated) will allow you to enable or disable Timed Automatic Outputs.

See Also: Defining Hot Keys and Hot Key Actions

In the bottom of the Output String Editor dialog box, you may also define up to 20 "Button Controlled Output Strings" (referenced as "String1" through "String20") that are each associated with a "button" in the Software Wedge window after you activate the Wedge. Clicking your mouse on an output string button in the Wedge window causes the string to be sent out the serial port to your device. When defining button controlled output strings you may also specify a "Button Caption" for each button to remind you of its contents or purpose. For example, many electronic balances can be prompted to transmit a weight reading by sending them a certain prompt string. For a Sartorius balance the prompt is an Escape character (ASCII 27) followed by a capitol P. For a Mettler balance the prompt is a capitol S followed by a carriage return (ASCII 13) and a linefeed (ASCII 10).

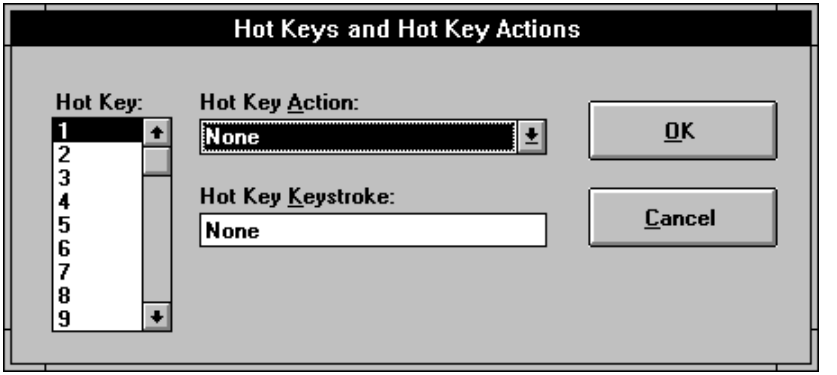
For this situation, you could define an output string containing the required prompt characters and also define its button caption as "Send Data". Then whenever you click your mouse on the button with the caption "Send Data" in the Software Wedge window, the prompt string would be sent to the scale causing it to send back a weight reading.

When defining a caption for a button you can assign an *access key* to the button by including an ampersand (&) in the caption immediately preceding the character you want to be used as the access key. This character will appear underlined in the button caption when the Wedge is activated. Pressing the underlined character in combination with the Alt key while the Software Wedge is active and has the input focus has the same effect as clicking your mouse on the button. For example, specifying "&Send Data" for a button caption would cause the button to appear as below with Alt + S being the access key.

When editing any of the output strings in the Output String Editor dialog box, an ASCII chart is available so that you can select ASCII characters or control codes that cannot be entered on your keyboard. For example to enter a specific ASCII character in an output string, move the cursor to the point in the output string where you would like the character to go and then click your mouse on the button marked "ASCII Chart". When the ASCII chart appears, scroll down the list until the character that you want is highlighted and click the OK button in the ASCII chart. The character will appear where the cursor is in the output string.

Defining Hot Keys and Hot Key Actions

The Software Wedge allows you to define up to 50 hot keys that can be used to control several serial I/O functions and also control the operation of the Wedge. When you select “Hot Keys and Hot Key Actions” from the Define menu the following dialog box appear.



Each hot key that you can define has a number from one to fifty and you can select a particular hot key by choosing its hot key number from the list box with the caption “Hot Key” in the upper left corner of the dialog box.

Selecting Hot Key Actions

You define a hot key by first selecting a *Hot Key Action* from the list of available actions and then you specify the keystroke that will be used to invoke each desired action. The Software Wedge supports the following 13 hot key actions:

	Hot Key Action	**Meaning**
1.	**Transmit String**	Transmits a character string out the serial port. When you select “Transmit String”, a text input box will appear in the bottom of the dialog box with the caption “Output String”. This is where you enter the character string that you want transmitted out the serial port when the hot key is pressed. When editing the output string, a button with the caption “ASCII Chart...” will also appear in the dialog box so that you can choose control codes (or ASCII characters that cannot be typed on your keyboard) that you want to place in the output string.
2.	**Issue BREAK Signal**	Issues a serial BREAK signal
3.	**Toggle DTR for 100ms**	Toggles the serial port DTR line for 100ms
4.	**Enable Timer**	Enables timer controlled outputs
5.	**Disable Timer**	Disables timer controlled outputs

See Also: Defining Serial Output Strings

6.	**Raise DTR**	Raises the serial port DTR line
7.	**Lower DTR**	Lowers the serial port DTR line
8.	**Raise RTS**	Raises the serial port RTS line
9.	**Lower RTS**	Lowers the serial port RTS line
10.	**Reset WinWedge**	Resets the Wedge. When you perform a reset, the serial input buffer is flushed and the Wedge is reset to the state it was in when first activated. See Also: Activating The Software Wedge
11.	**Suspend WinWedge**	Suspends the Wedge. While suspended, the Wedge continues to collect serial data and store it to a serial input buffer however the data will not be transferred to another program or to a disk file until you resume the Wedge.
12.	**Resume WinWedge**	Resumes the Wedge if it is currently suspended
13.	**Resume for 1 Data Record**	Resumes the Wedge for a single data record only

If the Wedge is suspended and you resume it for one data record, the next data record available in the serial receive buffer will be processed after which the Wedge will automatically suspend itself again. The Resume for 1 Data Record action thus allows a greater degree of control over where and when data is entered into an application because it lets you accept data specifically when you are ready to receive it.

Selecting Hot Key Keystrokes

To select a hot key keystroke that will invoke a particular hot key action, place the cursor in the text box labeled “Hot Key Keystroke” and then press the key or key combination that you want to use. A description of the keystroke will appear in the text box after you press the keystroke. When the cursor is in the “Hot Key Keystroke” text box, a button will also appear with the caption “Keystroke List...”. Clicking this button pops open a list box so that you can choose your hot key keystroke from the list.

Notes: Hot keys defined in the Wedge are system wide hot keys and thus are always detected by the Wedge no matter what application has the input focus. All hot key keystrokes are also discarded after they are detected by the Wedge and thus are not passed through to the application you are working in when you press the hot key. You may also define multiple hot key actions for the same hot key keystroke. If you do this, all actions assigned to the common hot key keystroke will be invoked when you press the hot key. In this situation hot key actions with lower hot key numbers will be invoked before those with higher numbers.

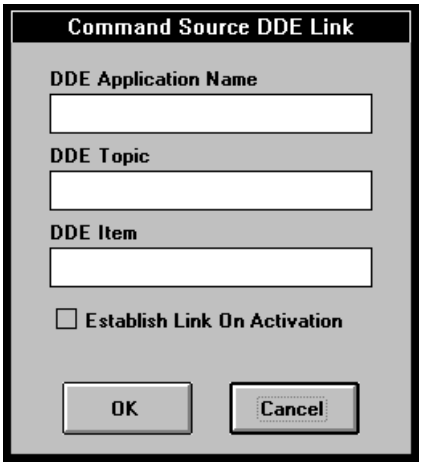
Tech Tip: When using the Wedge in “Send Keystrokes” mode, it is possible for the Wedge to send itself hot key keystrokes. This interesting side effect could be taken advantage of to cause the Wedge to suspend or reset itself if a certain character is received. (Use the Pre-Transfer Character Translation Table to translate the character to a hot key keystroke whose action is to suspend or reset the Wedge). You could also continuously poll an instrument by having the Wedge issue a “Field Postamble Keystroke Macro” that contains a hot key that invokes the “Transmit String” action.

Defining A Command Source DDE Link

The Software Wedge supports a set of commands that can be issued by other Windows programs to cause it to perform actions such as transmitting a data string out the serial port. Normally these "DDE commands" would be issued from within a script or macro language in another application (using the "DDEExecute" statement in Excel for example). Unfortunately, some applications that support DDE do not provide a way to issue DDE commands directly to other applications. The Command Source DDE Link option provides an alternative way for the Wedge to receive DDE commands from another application by allowing you to specify a source location or *DDE Server Item* in the other application that the Wedge will link to and use as the source for DDE commands.

Note: The "Command Source DDE Link" should only be used when the application that you are using does not provide any other way to send DDE commands.

When you select this option, the following dialog box appears:



The DDE Application Name, Topic and Item specify a DDE server item that the Wedge will link to and use as a source for its DDE commands. For example, the application name "EXCEL", topic "SHEET1" and item "R1C1" would specify that any data entered in cell R1C1 of the EXCEL spreadsheet SHEET1 should be acted upon as a command to the Wedge. The "Establish Link On Activation" check box causes the Wedge to automatically establish the DDE link as soon as it is activated. After you activate the Wedge with a Command Source DDE Link defined, it will establish a link to the specified DDE item and whenever the contents of the item changes, the Wedge will assume that the item contains a DDE command and will try to execute the command.

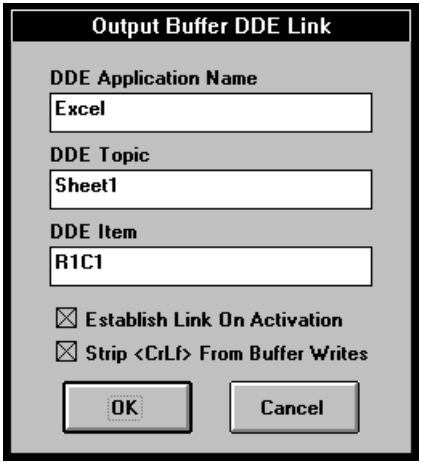
Note: Only valid Software Wedge DDE commands can be executed. Each command is executed once, when the contents of the source location changes. To issue the same command more than once, clear the source contents and replace it with the same command.

See Also: Software Wedge DDE Command Set

Defining An Output Buffer DDE Link

The *Output Buffer DDE Link* option in the "Define" menu opens a dialog box where you can specify a DDE *server* or source to be used as a serial output buffer by the Software Wedge. This option essentially allows you to place a serial output buffer directly in any other program that can act as a DDE server. For example you could specify a cell in a spreadsheet to be used as a serial output buffer such that whenever you write any data to the cell, the data would be automatically transmitted out the serial port by the Wedge.

When you select this option, the following dialog box appears:



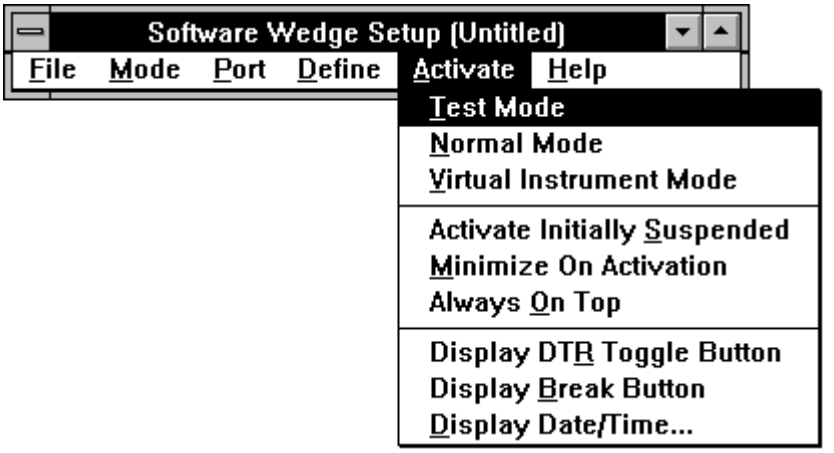
The DDE Application Name, Topic and Item specify a DDE server item that the Wedge will link to and use as a source for a serial output buffer. For example, the application name "EXCEL", topic "SHEET1" and item "R1C1" would specify that any data entered in the cell R1C1 of the EXCEL spreadsheet SHEET1 should be transmitted out the serial port by the Software Wedge. The "Establish Link On Activation" check box causes the Software Wedge to automatically establish the DDE link as soon as the Wedge is activated. The "Strip <CrLf> From Buffer Writes" check box causes the Wedge to remove a trailing carriage return - linefeed pair from each output data string. This option may be needed with some application programs like Microsoft Excel that automatically append a carriage return - linefeed to every data item written to a cell.

All serial output data from an Output Buffer DDE Link is transmitted once, when the contents of the source location changes. To transmit the same data more than once, clear the source contents and replace it with the same data.

Note: The Software Wedge provides a number of DDE commands that can be used to transmit data out the serial port. The "Output Buffer DDE Link" should only be used when the application that you are using does not provide a way to send DDE commands.

See Also: Software Wedge DDE Commands

The Activate Menu



The "Activate" menu contains nine entries. The first three entries, "Test Mode", "Normal Mode" and “Virtual Instrument Mode” are used to activate the Wedge for serial input. The Wedge must be activated using one of these options before it will actually perform any serial I/O functions or transmit data to an application program. These three options and the differences between them are described in detail in the section: Activating The Software Wedge

The next six items are "ON / OFF" options that are turned on or off by selecting them. When selected (ON), a check mark will appear to the left of each menu item.

The “Activate Initially Suspended” option causes the Software Wedge to be suspended when it is initially activated. When the Wedge is suspended, it will still collect serial data and store it in a buffer until you resume the Wedge either by selecting a “resume” option from the activated Wedge’s menu or by pressing a hot key or issuing a DDE command to WinWedge that causes it to resume its operation.

The next five options, "Minimize On Activation", “Always On Top”, "Display DTR Toggle Button", “Display Break Button” and "Display Date/Time..." control the appearance of the window that is displayed when the Software Wedge is activated (the Software Wedge Window).

The "Minimize On Activation" option causes the Software Wedge to run minimized or as an icon when activated. The Wedge will continue to function even when minimized.

The “Always On Top” option causes the Software Wedge Window to remain on top of all other windows after you activate it.

The "Display DTR Toggle Button" option causes a button to appear in the Software Wedge Window with the caption "DTR~". This button is used to toggle the serial port Data Terminal Ready line for 100 ms. This action is interpreted by certain types of serial devices as a request to transmit a data record.

The "Display Break Button" option causes a button to appear in the Software Wedge Window with the caption "BREAK". This button is used to issue a serial BREAK signal. Certain types of serial instruments use the Break signal as a request to either transmit a data record or perform a reset function. For example, most modems will hang up and reset themselves if a BREAK signal is issued to them.

The "Display Date/Time" option turns on a Date / Time display in the Software Wedge Window. If you enable this option, you will also be prompted to enter a format expression for the displayed Date and / or Time using formatting characters described in the section: "Formatting Date / Time Expressions". When the Wedge is activated with the Date/Time display enabled, a text box will appear in the Software Wedge Window that contains the Date or Time formatted using the expression that you specify. If no formatting expression is specified, a default of "mm/dd/yy hh:nn:ss AM/PM" is used. The purpose of the Date/Time display is to provide a real time clock that can be linked via DDE to other Windows programs. The DDE Application name for the Date/Time display is "WinWedge" the DDE Topic is "COMn" where n is the serial port (1-9) that the Wedge is currently activated for, and the DDE Item name is "DATETIME".

See Also: Understanding Dynamic Data
Establishing DDE Links Using the Windows Clipboard
DDE And The Software Wedge

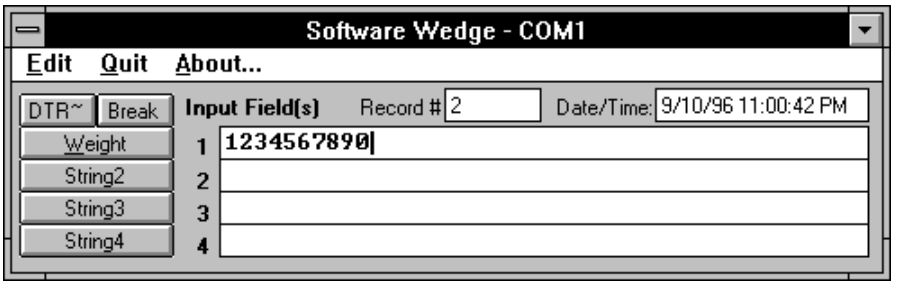
Activating the Wedge in Test Mode or Normal Mode

The Software Wedge will not accept any serial data until it is activated by selecting either “Test Mode” or “Normal Mode” from the Activate menu. The difference between the two modes is that when you activate the Wedge in "Test Mode", you will be able to return to the Software Wedge main menu and edit your configuration parameters. Test mode should always be selected when you are first configuring the Wedge for use with a particular application and a particular serial device. Until you have everything working properly, you will find it much easier to activate the Wedge in Test mode.

In "Normal Mode" you will not be able to go back and edit any configuration parameters without quitting the Wedge altogether and then restarting it and reloading your configuration file. When you activate the Wedge in "Normal Mode", its main menu and all sub menus, dialog boxes, and setup code are unloaded from memory in order to conserve Windows system resources. This helps free up system memory and other resources that may be needed by other application programs and it also improves overall system performance.

The Software Wedge Window

When you activate the Wedge in either Test or Normal mode, the main menu will disappear and a window similar to the one below will appear. This window is called the "Software Wedge Window".



The title bar of the Software Wedge Window will indicate the serial port that it is currently configured for.

The menu bar in the Software Wedge Window contains two entries, Edit and Quit.

The number of "Input Data Fields" (the text boxes under the label "Input Field(s)") and "Serial Output Buttons" (the column of buttons on the left side of the window) that will be visible in the Software Wedge Window depends on how many were previously defined when you configured the Software Wedge.

See Also: Defining Serial Output Strings

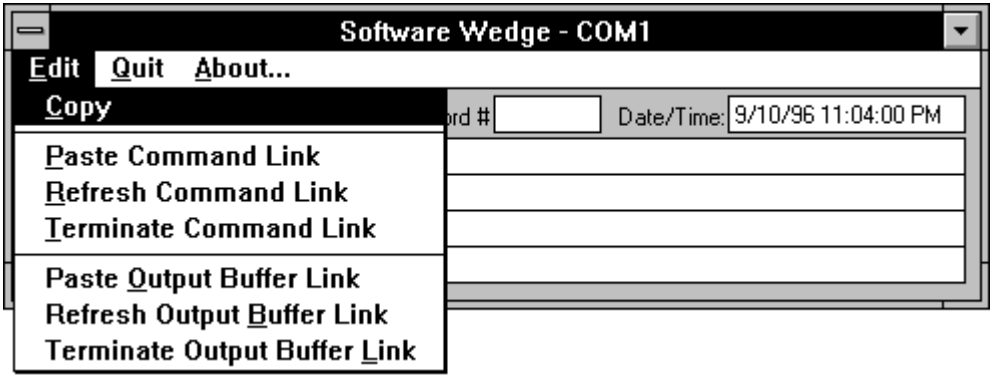
The button with the caption "DTR~" is called the "Data Terminal Ready Line Toggle Button". This button will appear only if the "Display DTR Toggle Button" option is selected in the Activate menu. Pressing this button causes the serial port Data Terminal Ready Line to be toggled from "on" to "off" for approx. 100 ms. This action is interpreted by certain types of serial devices as a request for data. Similar to the DTR toggle button, the “Break” button can be used to issue a BREAK signal to a device.

The “Record #” text box will display a count of the total number of input data records that have been processed since the Wedge was activated.

Note: The Software Wedge Window can be shrunk to an icon without affecting its operation. The only disadvantage of "minimizing" the Software Wedge Window is that you must re-open it in order to press any of the output string buttons. If you want to run the Wedge minimized and still be able to send data out the serial port, you would have to use other features in the Wedge such as Hot Keys or DDE Commands to transmit data.

The Edit Menu

The Edit menu contains the entries *Copy*, *Paste Command Link*, *Refresh Command Link*, *Terminate Command Link, Paste Output Buffer Link*, *Refresh Output Buffer Link* and *Terminate Output Buffer Link*.



The *Copy* command allows you to copy the contents of one of your input data fields, the record number or the date/time display into the Windows clipboard. To use this feature, click your mouse in the text box that you want to copy and select "Copy" from the menu. Any data copied from the Software Wedge can be either *Pasted* or *Paste Linked* to other Windows applications.

See Also: Establishing DDE Links Using the Windows Clipboard

The next three options, *Paste Command Link*, *Refresh Command Link*, *Terminate Command Link* allow you to establish, refresh and terminate DDE Command Links with other applications. The *Paste Command Link* option will be active (not grayed) if the clipboard contains text data that has been copied from another application that can act as a DDE Server (a cell in a spreadsheet for example).

A Command Source DDE Link can be pre-defined by selecting *Command Source DDE Link* from the main menu. Pasting a *Command Link* after the Software Wedge has been activated is simply another way to specify or change a Command Source DDE Link.

See Also: Defining A Command Source DDE Link
Software Wedge DDE Commands

The Refresh Command Link option causes the Software Wedge to attempt to re-establish the last Command Source DDE Link that was active. This option will be active only when a Command Source DDE Link has been previously defined either by explicitly defining one from the main menu *Command Source DDE Link* option or by *Pasting* a Command Link.

The option *Terminate Command Link* causes the Wedge to terminate any active Command Source DDE Link. This option will be available only when a Command Source DDE Link is currently active.

The last three options in the EDIT menu, *Paste Output Buffer Link*, *Refresh Output Buffer Link*, and *Terminate Output Buffer Link* allow you to establish, refresh and terminate DDE Output Buffer Links with other applications. The *Paste Output Buffer Link* option will be active (not grayed) if the clipboard contains text data that has been copied from another application that can act as a DDE Server (again, a cell in a spreadsheet for example). An Output Buffer DDE Link can also be pre-defined by selecting *Output Buffer DDE Link* from the main menu. Pasting an *Output Buffer Link* after the Software Wedge has been activated is another way to specify or change an Output Buffer DDE Link.

See Also: Defining An Output Buffer DDE Link.

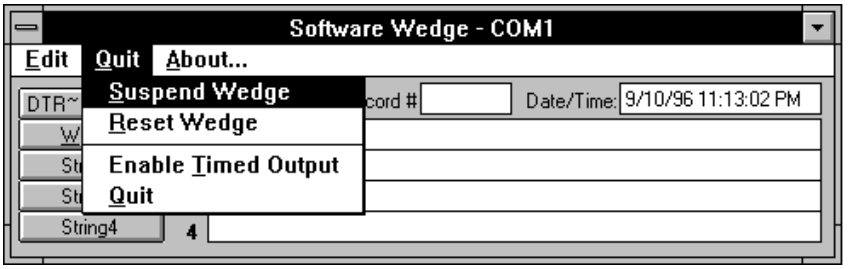
The Refresh Output Buffer Link option causes the Wedge to attempt to re-establish the last Output Buffer DDE Link that was active. This option will be active only when an Output Buffer DDE Link has been previously defined either by explicitly defining one from the main menu *Output Buffer DDE Link* option or by previously *Pasting* an Output Buffer Link.

The option *Terminate Output Buffer Link* causes the Software Wedge to terminate any active Output Buffer DDE Link. This option will be available only when an Output Buffer DDE Link is currently active.

The Quit Menu

The Quit menu provides three main choices, "Suspend", “Reset” and "Quit". If a timer controlled output string was defined, a fourth option "Enable/Disable Timed Output" will also appear.

See Also: Defining Serial Output Strings



Selecting "Suspend" suspends the operation of the Software Wedge and changes the Suspend menu item to "Resume" (which allows you to resume the Wedge). While the Wedge is suspended, the title bar will contain an S in parentheses (S) to remind you that the Wedge is currently suspended. When the Wedge is suspended, there will also be an additional Quit menu option with the caption “Resume For One Record” that allows you to resume the Wedge for a single data record after which the Wedge will be automatically re-suspended. This feature allows you to pull in data one record at a time.

See Also: Defining Hot Keys and Hot Key Actions
Software Wedge DDE Commands

If you select Reset from the Quit menu, the Software Wedge will flush its serial receive buffer and also reset all internal variables to their default values. This has the same effect as quitting and then reactivating the Wedge.

If you quit the Wedge after activating it in "Test Mode", you will be returned to the main menu. If the Wedge was originally activated in "Normal Mode" or by specifying a configuration file name on the command line used to launch the Wedge, the Wedge will quit running altogether.

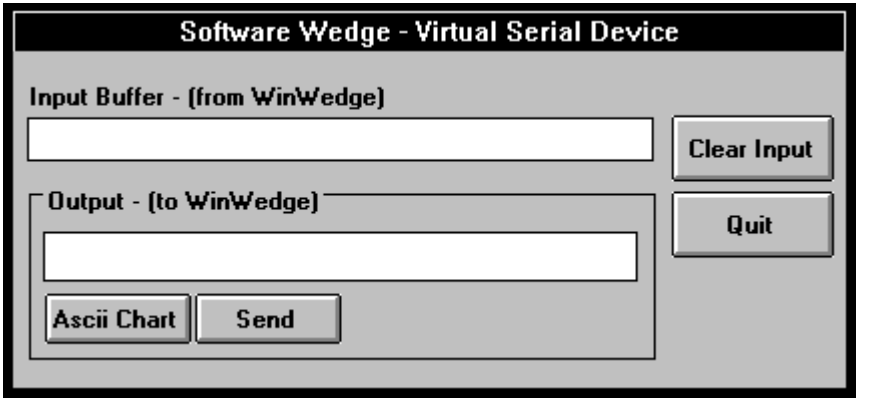
See Also: Activating The Wedge Automatically With A Specific Configuration File

The *Enable/(Disable) Timed Output* option is used to turn on and off any automatic timer controlled output that has been defined. If a timer controlled output has been defined and is currently Enabled, then this item will have the caption "Disable Timed Output". Likewise, if the timer controlled output string is currently Disabled, this item will have the caption "Enable Timed Output" thus allowing you to re-enable it.

See Also: Defining Serial Output Strings

Activating the Software Wedge in Virtual Instrument Mode

When you activate the Wedge in “Virtual Instrument mode”, it will behave exactly as it does when you activate it in “Test Mode” except that instead of communicating with a physical instrument connected to one of the serial ports on your PC, an additional dialog box called a "virtual serial device" dialog box will appear on your screen.



The virtual serial device dialog box allows you to test the functionality of the Wedge without requiring any real hardware device to be connected to a serial port on your PC. The dialog box has an “Input Buffer” that will display any data that would normally be transmitted by the Wedge out a serial port and it also provides an “Output Buffer” where you can enter and send data to the Wedge to simulate data that normally would be received from a device connected to an actual serial port.

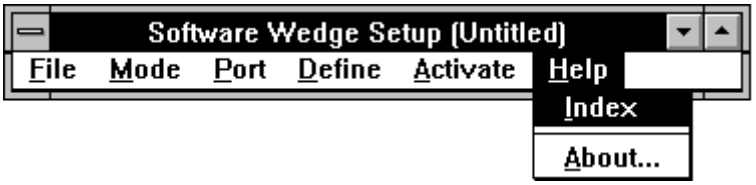
The virtual serial device dialog box thus allows you test the functionality of the Wedge without requiring an actual instrument or device to be connected to a physical serial port on your PC. To simulate a real device, simply enter some typical data in the “Output Buffer” and click on the button marked “Send”. This will transmit the data from the “Output Buffer” to the Wedge just as if it were being transmitted through a serial port. To enter control codes or other characters that cannot be typed on your keyboard into the Output Buffer, an ASCII chart is available by clicking the button marked “ASCII Chart”. Any data that is transmitted from the Wedge will appear in the Input Buffer.

The button marked “Clear Input” is used to clear out the contents of the Input Buffer text box and the button marked “Quit” will return you to the main menu of the Software Wedge.

Note: When you send data to the Wedge from the Virtual Serial Device dialog (and from a real instrument), the Software Wedge will not do anything with the data until the specified “End of Record Event” occurs. For example, if you specified the “End of Record Event” as a carriage return or a special character received, you should include these characters in any data that you transmit to the Wedge. You can enter a carriage return - linefeed pair (ASCII 13 & ASCII 10) in the Output Buffer by pressing the Enter key or you can select these characters from the ASCII chart.

See Also: Defining the Input Data Record Structure

Accessing On-Line Help



On-Line help is available from the main menu of the Software Wedge by selecting the Help menu item "Index". This option will access the Windows help system and display an index of available help topics for the Software Wedge. On-Line help is only available from the main (setup) menu and can not be accessed after the Software Wedge has been activated.

Context sensitive help is also available by pressing function key F1.

Selecting the sub menu item "About" will display a copyright notice.

Keystroke Macro Rules

When editing “Record Preamble” or “Field Postamble” keystroke macros with the Wedge in "Send Keystrokes Mode", to specify a single keyboard character, use the character itself. For example, to represent the letter A, type an "A". If you want to represent more than one character, append each character to the one before. To represent the letters a, b, and c, simply enter: abc. The plus sign (+), caret (^), percent sign (%), tilde (~) and parentheses () have special meanings to the Software Wedge. To specify one of these special characters, enclose the character inside curly braces. For example, to specify the plus sign, use {+}. Square brackets ([]) have no special meaning to the Software Wedge but you must enclose them in braces as well because square brackets have special meaning when performing DDE operations. To send curly brace characters, use "{{}" and "{}}", respectively. To specify characters that are not displayed when you press a key (such as Enter or Tab) and other keys that represent actions rather than characters, use the codes in the table below:

Key	Code	Key	Code
Backspace	{BACKSPACE}, {BS} or {BKSP}	Tab	{TAB}
Break	{BREAK}	Up	{UP}
CapsLock	{CAPSLOCK}	F1	{F1}
Clear	{CLEAR}	F2	{F2}
Delete	{DELETE} or {DEL}	F3	{F3}
Down	{DOWN}	F4	{F4}
End	{END}	F5	{F5}
Enter	{ENTER} or ~	F6	{F6}
Escape	{ESCAPE} or {ESC}	F7	{F7}
Help	{HELP}	F8	{F8}
Home	{HOME}	F9	{F9}
Insert	{INSERT}	F10	{F10}
Left	{LEFT}	F11	{F11}
NumLock	{NUMLOCK}	F12	{F12}
Page Down	{PGDN}	F13	{F13}
Page Up	{PGUP}	F14	{F14}
PrtScrn	{PRTSC}	F15	{F15)
Right	{RIGHT}	F16	{F16}
Scroll Lock	{SCROLLLOCK}		

To specify keys combined with any combination of Shift, Control, and Alt, precede the regular key code with one or more of these codes:

Key	Code
Shift	+
Control	^
Alt	%

To specify that Shift, Control, and/or Alt should be held down while several keys are pressed, enclose the keys in parentheses. For example, to hold the Shift key while pressing E then C, use "+(EC)". To hold down Shift while pressing E, followed by C without the Shift key, use "+EC". To specify repeating keys, use the form {key number} where there is always a space between key and number. For example, {LEFT 42} means press the “Left Arrow” key 42 times; {x 10} means press the letter "x" 10 times.

Date And Time Stamps In Macros

The Software Wedge supports six date and time stamp functions that are specified by entering the following keywords in any preamble or postamble macro defined when the Software Wedge is in *Send Keystrokes Mode* or *Log To Disk Mode* .

Keyword	Function	Range
{Century}	Inserts the current century	19 - 99
{Year}	Inserts the current year	00 - 99
{Month}	Inserts the current month	01 - 12
{Day}	Inserts the current day	01 - 31
{Hour}	Inserts the current hour	00 - 24
{Minute}	Inserts the current minute	00 - 59
{Second}	Inserts the current second	00 - 59

All date & time values are sent as a two byte string padded with a zero on the left if the value is less than ten. Only one occurrence of each stamp function may be specified in any one macro. For example, the following macro: "{Year}/{Month}/{Day}:{Hour}:{Minute}:{Second}" is valid but "{Year}{Year}" is invalid.

Format Expressions

When defining output data formats for each field of data in an input record, format expressions can be applied to either numbers or strings using special formatting characters contained in the format expression.

Formatting Numbers

To format numbers, you can use one of the commonly-used formats that have been pre-defined or you can create user-defined formats with standard characters that have special meaning when used in a format expression. The following table shows the pre-defined numeric format names you can use and the meaning of each:

Format name	Description
General Number	Display the number as is, with no thousand separators.
Currency	Display number with thousand separator, if appropriate; display negative numbers enclosed in parentheses; display two digits to the right of the decimal separator.
Fixed	Display at least one digit to the left and two digits to the right of the decimal separator.
Standard	Display number with thousand separator, if appropriate; display two digits to the right of the decimal separator.
Percent	Display number multiplied by 100 with a percent sign (%) appended to the right; display two digits to the right of the decimal separator.
Scientific	Use standard scientific notation.
Yes/No	Display *No* if number is 0, otherwise display *Yes*.
True/False	Display *False* if number is 0, otherwise display *True*.
On/Off	Display *Off* if number is 0, otherwise display *On*.

The following table shows the characters you can use to create user-defined number formats and the meaning of each:

Character	Meaning
0	Digit placeholder. Display a digit or a zero. If there is a digit in the expression being formatted in the position where the 0 appears in the format string, display it; otherwise, display a zero in that position. If the number being formatted has fewer digits than there are zeros (on either side of the decimal) in the format expression, leading or trailing zeros are displayed. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, the number is rounded to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, the extra digits are displayed without modification.
#	Digit placeholder. Display a digit or nothing. If there is a digit in the expression being formatted in the position where the # appears in the format string, display it; otherwise, display nothing in that position. This symbol works like the 0 digit placeholder, except that leading and trailing zeros aren't displayed if the number has the same or fewer digits than there are # characters on either side of the decimal separator in the format expression.
.	Decimal placeholder. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only number signs to the left of this symbol, numbers smaller than 1 begin with a decimal separator. If you want a leading zero to always be displayed with fractional numbers, use 0 as the first digit placeholder to the left of the decimal separator instead. The actual character used as a decimal placeholder in the formatted output depends on the Number Format specified in the International section of the Microsoft Windows Control Panel. For some countries, a comma is used as the decimal separator.

% Percentage placeholder.
The expression is multiplied by 100. The percent character (%) is inserted in the position where it appears in the format string.

, Thousand separator.
The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a comma surrounded by digit placeholders (0 or #). Two adjacent commas or a comma immediately to the left of the decimal separator (whether or not a decimal is specified) means "scale the number by dividing it by 1000, rounding as needed". You can scale large numbers using this technique. For example, you can use the format string "##0,," to represent 100 million as 100. Numbers smaller than 1 million are displayed as 0. Two adjacent commas in any position other than immediately to the left of the decimal separator are treated simply as specifying the use of a thousand separator. The actual character used as the thousand separator in the formatted output depends on the Number Format specified in the International section of the Control Panel. For some countries, a period is used as the thousand separator.

E- E+ e- e+ Scientific format.
If the format expression contains at least one digit placeholder (0 or #) to the right of E-, E+, e-, or e+, the number is displayed in scientific format and E or e is inserted between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a minus sign next to negative exponents and a plus sign next to positive exponents.

- + $ ( ) space Display a literal character.
To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks (" ").

**** Display the next character in the format string.
Many characters in the format expression have a special meaning and can't be displayed as literal characters unless they are preceded by a backslash. The backslash itself isn't displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes (\\).

Examples of characters that can't be displayed as literal characters are the date- and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, and /:), the numeric-formatting characters (#, 0, %, E, e, comma, and period) and the string-formatting characters (@, &, <, > and !).

"ABC" Display the string inside the double quotation marks.

***** Display the next character as the fill character.
Any empty space in a field is filled with the character following the asterisk.

Unless the format expression contains one of the pre-defined formats, a format expression for numbers can have from one to four sections separated by semicolons.

If you use	**The result is**
One section only	The format expression applies to all values.
Two sections	The first section applies to positive values and zeros, the second to negative values.
Three sections	The first section applies to positive values, the second to negative values and the third to zeros.
Four sections	The first section applies to positive values, the second to negative values, the third to zeros and the fourth to Nul values.

The following example has two sections: the first defines the format for positive values and zeros; the second section defines the format for negative values.

```
"$#,##0;($#,##0)"
```

If you include semicolons with nothing between them, the missing section is printed using the format of the positive value. For example, the following format displays positive and negative values using the format in the first section and displays "Zero" if the value is zero.

```
"$#,##0;;\Z\e\r\o"
```

Some sample format expressions for numbers are shown below. (These examples all assume that Country is set to United States in the International section of the Control Panel). The first column contains the format strings. The other columns contain the output that results if the formatted data has the value given in the column headings.

Format	**Positive 5**	**Negative 5**	**Decimal .5**	**Nul**
Nul string	5	-5	0.5	
0	5	-5	1	
0.00	5.00	-5.00	0.50	
#,##0	5	-5	1	
#,##0.00;;;Nil	5.00	-5.00	0.50	Nil
$#,##0;($#,##0)	$5	($5)	$1	
$#,##0.00;($#,##0.00)	$5.00	($5.00)	$0.50	
0%	500%	-500%	50%	
0.00%	500.00%	-500.00%	50.00%	
0.00E+00	5.00E+00	-5.00E+00	5.00E-01	
0.00E-00	5.00E00	-5.00E00	5.00E-01	

Formatting Character Strings

Character strings can also be formatted using special formatting characters. A format expression for strings can have one section or two sections separated by a semicolon.

If you use	The result is
One section only	The format applies to all string data.
Two sections	The first section applies to string data, the second to Nul values and zero-length strings.

You can use any of the following characters to create a format expression for strings:

Character	Meaning
@	Character placeholder. Display a character or a space. If there is a character in the string being formatted in the position where the @ appears in the format string, display it; otherwise, display a space in that position. Placeholders are filled from right to left unless there is an ! character in the format string. See below.
&	Character placeholder. Display a character or nothing. If there is a character in the string being formatted in the position where the & appears, display it; otherwise, display nothing. Placeholders are filled from right to left unless there is an ! character in the format string. See below.
<	Force lowercase. All characters are displayed in lowercase format.
>	Force uppercase. All characters are displayed in uppercase format.
!	Force placeholders to fill from left to right instead of right to left.

Formatting Date/Time Expressions

Date/Time format expressions can be used to format the Date/Time display in the Software Wedge window and can also be used with the SENDDATE(format) DDE command.

See Also:
The Activate Menu
Software Wedge DDE Commands

To format dates and times, you can specify either the name of a commonly used format that has been pre-defined in the Software Wedge or you can create user-defined date/time formats using special formatting characters that have special meaning when used in a format expression. The following table shows the pre-defined data format names you can use and the meaning of each:

Format Name	**Description**
General Date	Display a date and/or time. For real numbers, display a date and time. (e.g. 4/3/93 05:34 PM); If there is no fractional part, display only a date (e.g. 4/3/93); if there is no integer part, display time only (e.g. 05:34 PM).
Long Date	Display a Long Date, as defined in the International section of the Control Panel.
Medium Date	Display a date in the same form as the Short Date, as defined in the International section of the Control Panel, except spell out the month abbreviation.
Short Date	Display a Short Date, as defined in the International section of the Control Panel.
Long Time	Display a Long Time, as defined in the International section of the Control Panel. Long Time includes hours, minutes, seconds.
Medium Time	Display time in 12-hour format using hours and minutes and the AM/PM designator.
Short Time	Display a time using the 24-hour format (e.g. 17:45)

The following table shows the characters you can use to create user-defined date/time formats and the meaning of each:

Character	Meaning
:	Time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator depends on the Time Format specified in the International section of the Control Panel.
/	Date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in the formatted output depends on Date Format specified in the International section of the Control Panel.
c	Display the date as ddddd and display the time as t t t t t, in that order. Only date information is displayed if there is no fractional part to the date serial number; only time information is displayed if there is no integer portion.
d	Display the day as a number without a leading zero (1-31).
dd	Display the day as a number with a leading zero (01-31).
ddd	Display the day as an abbreviation (Sun-Sat).
dddd	Display the day as a full name (Sunday-Saturday).
ddddd	Display a date serial number as a complete date (including day, month and year) formatted according to the Short Date setting in the International section of the Windows Control Panel. The default Short Date format is m/d/yy.
dddddd	Display a date serial number as a complete date (including day, month and year) formatted according to the Long Date setting in the International section of the Control Panel. The default Long Date format is mmmm dd, yyyy.
w	Display the day of the week as a number (1 for Sunday through 7 for Saturday).

ww	Display the week of the year as a number (1-53).
m	Display the month as a number without a leading zero (1-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mm	Display the month as a number with a leading zero (01-12). If m immediately follows h or hh, the minute rather than the month is displayed.
mmm	Display the month as an abbreviation (Jan-Dec).
mmmm	Display the month as a full month name (January-December).
q	Display the quarter of the year as a number (1-4).
y	Display the day of the year as a number (1-366).
yy	Display the year as a two-digit number (00-99).
yyyy	Display the year as a four-digit number (100-9999).
h	Display the hour as a number without leading zeros (0-23).
hh	Display the hour as a number with leading zeros (00-23).
n	Display the minute as a number without leading zeros (0-59).
nn	Display the minute as a number with leading zeros (00-59).
s	Display the second as a number without leading zeros (0-59).
ss	Display the second as a number with leading zeros (00-59).
ttttt	Display a time serial number as a complete time (including hour, minute and second) formatted using the time separator defined by the Time Format in the International section of the Control Panel. A leading zero is displayed if the Leading Zero option is selected and the time is before 10:00 A.M. or P.M.. The default time format is h:mm:ss.

AM/PM Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 PM.

am/pm Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 PM.

A/P Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 PM.

a/p Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 PM.

AMPM Use the 12-hour clock and display the contents of the 1159 string (s1159) in the WIN.INI file with any hour before noon; display the contents of the 2359 string (s2359) with any hour between noon and 11:59 PM. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as it exists in the WIN.INI file. The default format is AM/PM.

The following are examples of user-defined date and time formats:

Format	Display
m/d/yy	12/7/58
d-mmmm-yy	7-December-58
d-mmmm	7 December
mmmm-yy	December 58
hh:mm AM/PM	08:50 PM
h:mm:ss a/p	8:50:35 p
h:mm	20:50
h:mm:ss	20:50:35
m/d/yy h:mm	12/7/58 20:50

Mathematical Functions and Operators

When defining Math Expressions in the “Input Record Definition Editor” dialog box, you may use any of the following functions or operators as well as numeric or string constants.

General Functions

Field(n) Returns the value of data field n from the Software Wedge.
The Field(n) function references data the *Input Data Fields* in the Software Wedge window. The Field(n) function can only reference either the data field where the math expression is used or lower numbered fields. For example, when defining a math expression for Field(2), you may use the functions “Field(1)” or “Field(2)” in the expression but you cannot use the function “Field(3)”.
For example, to change the value of Field(1) to the square root of Field(1) times 10, you would enter: 10 * SQR(Field(1)) as the math expression for Field(1).

Asc(s) Returns the ASCII value of the first character in the string s
UCase(s) Returns the string argument s in all upper case.
LCase(s) Returns the string argument s in all lower case.
LTrim(s) Removes leading spaces from the string argument s.
RTrim(s) Removes trailing spaces from the string argument s.
Trim(s) Removes leading and trailing spaces from the argument s.

Math Functions

Int(n) Returns the integer portion of the expression n
Fix(n) Returns the integer portion of the expression n

The difference between Int and Fix is that if number is negative, Int returns the first negative integer less than or equal to number, whereas Fix returns the first negative integer greater than or equal to number. For example, Int converts -8.4 to -9, and Fix converts -8.4 to -8.

Rnd(n) Returns a random number
The argument n can be any valid numeric expression.
The Rnd function returns a Single value less than 1 but greater than or equal to 0. The value of n determines how Rnd generates a random number:

Value of n	Number returned
< 0	The same number every time, as determined by number.
> 0	The next random number in the sequence.
= 0	The number most recently generated.

The same random-number sequence is generated every time the Wedge is run because each successive call to the Rnd function uses the previous random number as a seed for the next number in the random-number sequence. To produce random integers in a given range, use this formula: Int((ubound - lbound + 1) * Rnd + lbound). Here, ubound is the highest number in the range, and lbound is the lowest number in the range.

Abs(n)	Returns the Absolute value of expression n
Hex(n)	Returns a string representing the hexadecimal value of a decimal number
&h(n)	Converts a hexadecimal number n to a decimal number.
Sgn(n)	Returns the sign of expression n (returns either 1, -1 or 0)
Sqr(n)	Returns the square root of expression n
Val(n)	Returns the value of expression n
Sin(n)	Returns the Sine of the angle n
Cos(n)	Returns the Cosine of the angle n
Tan(n)	Returns the Tangent of the angle n
Atn(n)	Returns the Arctangent of the number n
Exp(n)	Returns the e raised to the power n
Log(n)	Returns the natural logarithm of expression n

The argument n can be any valid numeric expression that results in a value greater than 0. The natural logarithm is the logarithm to the base e. The constant e is approximately 2.718282. You can calculate base-n logarithms for any number x by dividing the natural logarithm of x by the natural logarithm of n as follows: Logn = Log(x) / Log(n)

Math Operators

^	Raises a number to the power of an exponent
/	Division Operator
*****	Multiplication Operator
****	Integer Division Operator
mod	Divides two numbers and returns only the remainder
+	Addition Operator
-	Subtraction or Negation Operator

Comparison Operators

Operator	Meaning	True (-1) if	False (0) if
<	Less than	expr1 < expr2	expr1 >= expr2
<=	Less than or equal to	expr1 <= expr2	expr1 > expr2
>	Greater than	expr1 > expr2	expr1 <= expr2
>=	Greater than or equal to	expr1 >= expr2	expr1 < expr2
=	Equal to	expr1 = expr2	expr1 <> expr2
<>	Not equal to	expr1 <> expr2	expr1 = expr2

Logical Operators

not	Performs a logical negation operation on two expressions
and	Performs a logical conjunction operation on two expressions
or	Performs a logical disjunction operation on two expressions
xor	Performs a logical exclusion operation on two expressions
eqv	Performs a logical equivalence operation on two expressions
imp	Performs a logical implication operation on two expressions

Operator Precedence in Math Expressions

When several operations occur in an expression, each part is evaluated and resolved in a predetermined order. That order is known as operator precedence. Parentheses can be used to override the order of precedence and force some parts of an expression to be evaluated before others. Operations within parentheses are always performed before those outside. Within parentheses, however, normal operator precedence is maintained.

When expressions contain operators from more than one category, arithmetic operators are evaluated first, comparison operators are evaluated next, and logical operators are evaluated last. Within individual categories, operators are evaluated in the order of precedence shown below:

Arithmetic	**Comparison**	**Logical**
Exponentiation (^)	Equality (=)	Not
Negation (-)	Inequality (<>)	And
Multiplication and division (*, /)	Less than (<)	Or
Integer division (\)	Greater than (>)	Xor
Modulo arithmetic (Mod)	Less than or Equal to (<=)	Eqv
Addition and subtraction (+, -)	Greater than or Equal to (>=)	Imp

Note: All comparison operators have equal precedence, that is, they are evaluated in the left-to-right order in which they appear.

When multiplication and division occur together in an expression, each operation is evaluated as it occurs from left to right. Likewise, when addition and subtraction occur together in an expression, each operation is evaluated in order of appearance from left to right.

Understanding Dynamic Data Exchange

Dynamic Data Exchange is a feature of Windows that allows two programs to share data or send commands directly to each other. DDE can be thought of as a direct conversation between two application programs. In most cases, one application is providing some form of data (either text or graphics) to another application. The application that is the source of the data is called the "server" and the application that is receiving the data is called the "client". Thus, the Software Wedge is primarily a DDE Server that "serves" incoming serial data to other (client) programs. The Wedge can also act as a DDE client in some special cases.

Each data item that a server application can provide has a unique identifier consisting of three parts, a *DDE Application Name*, A *DDE Topic*, and a *DDE Item name.*
The DDE Application Name is almost always the executable filename for the server application (without the .EXE extension), the DDE Topic typically identifies a group or category of data in the server application and each data item that a server application can provide has a unique DDE Item name. Thus, the Application Name, Topic, and Item Name identify the exact source of the data in a server application that is to be linked. (Sort of like a telephone number with the three parts: Area Code, Exchange and Number.)

DDE links are always initiated in the client application. The client initiates a DDE link by broadcasting a message containing a *DDE Application Name*, A *DDE Topic*, and optionally a *DDE Item* to all other applications currently running. If a server application is running that can provide the data, it responds to the DDE initiate and the Windows operating system opens a "link" between the two applications. Fortunately, most Windows programs that support DDE insulate the user from the low level details of establishing DDE links and simply allow you to specify the Application Name, Topic, and Item Name for a specific piece of data and the link is then automatically established for you by your application program.

For example, if you enter the formula: =WinWedge|Com1!'Field(1)' in a cell in an Excel spreadsheet and then press the Enter key, Excel will automatically establish a DDE link between the Software Wedge and the spreadsheet cell. (Note: WinWedge must be running and activated for COM1 in order for the above link formula to be successful). After the link is established, any data in the text box "Field(1)" in the Software Wedge window will automatically appear in the "linked" cell in the spreadsheet. Also, whenever the data for Field(1) in the Wedge changes, the contents of the spreadsheet cell will automatically be updated with the new data. It is like having the operating system do an automatic cut & paste from the server to the client whenever the server's data changes. The formula: =WinWedge|Com1!'Field(1)' contains the three parts necessary to successfully link to the Software Wedge; the DDE Application Name (WinWedge), the DDE Topic (Com1) and the specific DDE Item Name 'Field(1)'.

Either application involved in a DDE conversation can terminate the link. Some applications have menu options that allow you to selectively terminate any open DDE links. Closing either of the linked applications also causes all links between the two programs to be terminated.

DDE also allows a client application to send commands to a server. The types of DDE commands, if any, that a server program can accept will vary depending on the application and should be well documented in the application's user's manual along with the DDE Application Name and Topic required by client programs to establish the DDE link to the server's command processor. The Software Wedge supports over three dozen DDE commands that allow other applications to control the Wedge or transmit data out a serial port. All of these commands are described in detail in the section: Software Wedge DDE Commands.

Establishing DDE Links Using The Windows Clipboard

Applications that support DDE usually provide an easy way to initiate DDE conversations using the Windows clipboard. If an application can function as a DDE client, it will almost always have a "Paste Link" or "Paste Special" command in its main menu (usually in an "Edit" menu). DDE server applications will likewise have a "Copy" command in their main menu (also in an "Edit" menu). Note: The presence of a "Copy" command in a program does not necessarily mean the program can act as a DDE server.

To initiate a DDE conversation, you would open up the **server** application and select the data to be linked using your mouse (the contents of an Input Data Field text box in the Software Wedge Window for example) and then "Copy" the data to the clipboard by selecting "Copy" in its Edit menu. Next, you would open up the **client** application and click your mouse in the position where you would like the "Linked" data to appear (a cell in a spreadsheet for example) and then choose "Paste Link" or "Paste Special" from the client application's Edit menu.

If the Cut & Paste Link process is successful, the data from the server will appear in the client application from that point on until the link is terminated as the result of ending either the client or the server application. That's all there is to it! No messing about with Application Names, Topics or Items because all that information is hidden away and passed implicitly through the Windows clipboard

Note: When using the Cut & Paste Link method, some client applications may display a dialog box that allows you to specify either a "Hot" or "Cold" link. A "Hot" link means that changes in the data from the server are immediately updated in the client when the change occurs. A "Cold" link means the linked data is updated only at the request of the client program, usually with some sort of "Request" command. Cold links are useful when the linked item consists of a huge amount of data, a large bitmap for example, because it is very inefficient to continually pass an entire bitmap graphic from a server to a client each time a single pixel changes in the server.
For small character based data items (as provided by the Software Wedge), you would normally specify a "Hot" link. In some programs a Hot link may also be referred to as an "Auto Update" link.

DDE and the Software Wedge

The Software Wedge can function as both a server and a client in conversations with other applications. As a server, the Software Wedge can supply data from the "Input Data Fields" and the Date/Time or Record# display fields in the Software Wedge window. It can also process DDE commands that are sent to it from other applications.

As a client, the Wedge can receive data or commands through an Output Buffer DDE Link and a Command Source DDE Link . It can also issue commands to other applications that support a DDE command language. DDE commands can only be sent from the Wedge to another application as a "Record Preamble DDE Command" or as "Field Postamble DDE Commands" when it is in DDE Server Mode. Because the Wedge is the client in these conversations, it needs to know the DDE Application and Topic for the Server program that it will be receiving data from or sending commands to. This is why the Wedge prompts you for a DDE Application Name and DDE Topic when you select “DDE Server” from the Mode menu.

See Also: Specifying DDE Commands (DDE Server Mode)
Defining A Command Source DDE Link
Defining An Output Buffer DDE Link

Other applications can initiate DDE links with the Software Wedge using the DDE Application Name "**WinWedge**" and the DDE Topic "**COMn**" where "n" is the number of the serial adapter that the Wedge is activated for. The DDE Items available will be the contents of each defined input data field and the contents of the Date/Time and Record# display in the Software Wedge window. The DDE Item names for the data fields are referenced as **"Field(1)"** through **"Field(n)"** where "n" is the highest defined field number. The DDE Item name for the Date/Time display is "**DATETIME**" and the DDE Item name for the Record# display field is "**RECORDNUMBER**". If you use the "Copy" & "Paste Link" method to initiate DDE conversations with the Software Wedge, this information will be automatically supplied to the client application and does not need to be explicitly specified.

The Software Wedge also has two hidden DDE Items with the names "**OutputBuffer**" and "**CommandBuffer**" that another application can use to "POKE" serial output data or DDE commands to the Wedge. DDE "Poking" is a process where a DDE client application sends data back to a DDE server item. Poking data to the OutputBuffer item would transmit the poked data out the serial port. Poking to the CommandBuffer performs the same function as sending a DDE Command to the Wedge.

To initiate DDE conversations with the Wedge from a macro in a spreadsheet or other application, you will have to refer to the user’s manual or on-line help for the specific application to learn the syntax of the commands or macro instructions that can be used to initiate DDE links, send DDE Commands, perform DDE Pokes or perform DDE Requests. Many example are provided in the "DDE Examples" section of this manual and should therefore provide a good overview of how to communicate with the Wedge via DDE.

See Also: DDE Examples.

Software Wedge DDE Commands

The Software Wedge supports all of the DDE commands listed below.
Note: DDE commands will only execute if the Software Wedge has been activated.

See Also : Activating The Software Wedge

There are two ways to issue DDE commands to the Wedge; either by issuing them directly from another application's macro or programming language or you can establish a "Command Source DDE Link" between another application and the Wedge and then issue commands by entering the command in the DDE Command source location (a cell in a spreadsheet for example).

See Also: Understanding Dynamic Data Exchange
Defining A Command Source DDE Link
DDE Examples

Command	Function
[APPEXIT]	Exits the Software Wedge.
[APPMINIMIZE]	Minimizes the Software Wedge Window.
[APPNORMAL]	Opens the Software Wedge Window.
[BEEP]	Causes the Software Wedge to Beep once.
[BREAK]	Issues a Serial Break signal.
[CLEAR]	Clears all input data fields in the Software Wedge Window.
[COPYFIELD(n)]	Copies the data field "n" into the Windows clipboard.
[COPYDATE]	Copies the date/time display field into the clipboard.
[COPYRECNUM]	Copies the record number display field into the clipboard.
[SUSPEND]	Suspends the Software Wedge.
[RESUME1]	Resumes the Software Wedge for one data record.
[RESUME]	Resumes the Software Wedge.
[RESET]	Flushes all buffers and resets the Software Wedge.
[RTS=ON]	Sets the serial port RTS line on.
[RTS=OFF]	Sets the serial port RTS line off.
[DTR=ON]	Sets the serial port DTR line on.
[DTR=OFF]	Sets the serial port DTR line off.
[TOGGLEDTR]	Toggles the serial port DTR line for 100 ms.
[SEND(text)]	Sends the text in parentheses out the serial port.
[SENDCODE(n)]	Sends the character with ASCII code "n" out the serial port.
[SENDDATE(format)]	Sends the date or time out the serial port using a specified Date/Time format string.
[SENDFILE(filename)]	Sends the contents of the specified file out the serial port.

[SENDOUT('text',n...)] Sends text and control codes out the serial port.
Text must be supplied as quoted strings using single quotes and control codes must be supplied as ASCII values. Text and control codes may be combined in any combination with individual elements separated by commas. For Example, [SENDOUT('Test',13,10)] sends the word *Test* followed by a carriage return (ASCII 13) and a line feed (ASCII 10).

[PUSH(button caption)] Pushes a button in the Software Wedge Window.
The button is specified using its caption. The specified caption should be the same as that entered when it was originally defined including ampersands.
See Also: Defining Serial Output Strings

[CONFIG(filename)] Reconfigures the Software Wedge.
filename specifies the configuration file to use and must include the filename extension and the full drive and directory path if it is not in the current directory. You may not specify a configuration that uses a different serial port than the one currently in use.

[STRIPCRLF-ON]
[STRIPCRLF-OFF] Enables / Disables stripping carriage return - linefeeds from data supplied through an Output Buffer DDE Link.

[TIMER-ON]
[TIMER-OFF] Enables / Disables sending of timed automatic output strings.

[TIMERINTERVAL=n] Changes the timing interval (ms) for timed automatic output strings to the value n (between 1 and 9,999,999).

[PASTE-COMMAND] Pastes a DDE Command Link using data in the clipboard.
See Also: Defining A Command Source DDE Link

[REFRESH-COMMAND] Refreshes the last DDE Command Link defined.

[TERMINATE-COMMAND] Terminates any active DDE Command Link .

[PASTE-BUFFER] Pastes an Output Buffer Link using data in the clipboard.
See Also: Defining An Output Buffer DDE Link

[REFRESH-BUFFER] Refreshes the last Output Buffer DDE Link defined.

[TERMINATE-BUFFER] Terminates any active Output Buffer DDE Link.

Note: All DDE commands must be enclosed in square brackets.

When sending DDE commands to the Wedge from another program, use the DDE Application Name "WinWedge" and the DDE Topic "COMn" where "n" is the number of the serial port that the Wedge has been activated for.

Using the LINKTEST Utility to test DDE Commands

The Software Wedge comes with a utility called LINKTEST that allows you to test all Software Wedge DDE commands or test DDE commands that can be sent to other programs. To test any Software Wedge DDE commands, the Wedge must be running and activated in either "Test Mode", "Normal Mode" or “Virtual Instrument Mode”.

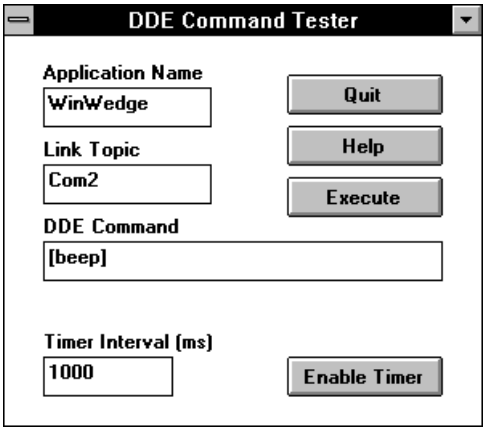
LINKTEST provides two text boxes where you specify the Application Name and DDE Link Topic for the application that you want to send a DDE command to. The defaults for these parameters are those required by the Software Wedge when it is activated for COM2. If you are using a different serial port, you should change the Link Topic to the correct value.

A third text box labeled "DDE Command" is where you enter the DDE Command that you want to send. After entering a DDE Command, you can have LINKTEST issue the command to the specified application by clicking the button marked "Execute".

If you get an error message that reads “No Foreign Application Responded to DDE Initiate” then you either did not specify the correct parameters for the DDE Application name or DDE Topic or the application is not currently running. If you get an error that reads “Foreign Application Won’t Perform DDE Method or Operation” then the application does not recognize either the command or the syntax for the command you are trying to have it execute.

For additional information about using LINKTEST including a list of all DDE Commands that the Software Wedge can recognize, click the button marked "Help" in the Link Test window.

If you use LINKTEST with an application other than the Software Wedge, you will have to refer to its users manual to find out the proper DDE parameters (Application Name and Topic) and also the proper syntax of any DDE commands that it can recognize.



Tech Notes

Fixing a problem in the Windows DDEML.DLL Dynamic Link Library

Background: There is a problem in the Windows DDEML.DLL dynamic link (provided as part of Windows) that causes the Software Wedge to not receive DDE commands sent to it in certain situations. For example if you configure the Wedge in DDE Server mode and define a "Field Postamble DDE Command" to be sent to another application to force it to run a macro after each new serial data record is received by the Wedge and the macro that you are forcing to run in the other program sends a DDE command back to the Wedge, the Wedge never gets the command and thus does not perform the desired function. The problem only occurs when a DDE command is sent to the Wedge from within a subroutine that was forced to run by a DDE command sent from the Wedge. Below is an example of a Microsoft Excel macro that is typical in this situation:

```
Sub GetSWData()          ' Field Postamble DDE Command from WinWedge runs this sub
If myRow = 0 Then myRow = 1                          ' make sure myRow not zero
chan = DDEInitiate ("WinWedge","COM1")                    ' open a DDE channel
myData = DDERequest(chan,"Field(1)")                 ' request data from Wedge
Sheets("Sheet1").Cells(myRow,1).Formula = myData     ' put the data in the sheet

DDEExecute chan, "[Sendout('bla bla bla')]"          ' send a command to Wedge
' WinWedge never gets the DDE command above due to a bug in DDEML.DLL

DDETerminate chan                                    ' terminate the dde link
myRow = myRow + 1                                    ' point to next row down
End Sub
```

How To Fix The Problem:

The problem described above is easily corrected by placing the following command anywhere in your AUTOEXEC.BAT file and re-booting your PC. (Windows 3.1 and Windows 95/98 & NT)

SET SRDDE=aoxomoxoa

Setting the above environment variable changes the behavior of the DDEML library so that the problem described above disappears. After extensive testing, we have not been able to find any negative side effects from setting the above environment variable therefore we highly recommend adding it to your AUTOEXEC.BAT file, even if you never expect to encounter the problem described here.

Microsoft makes no mention of this bug in the DDEML library anywhere in their entire tech knowledge base. The solution above was obtained from a senior support rep at Microsoft who described the problem as an "undocumented feature" of the Windows DDEML library. Interestingly enough any program that does not use the DDEML library does not experience this "undocumented feature".

Understanding How DDE Works Helps Avoid Common Programming Mistakes

Windows is a "message based", multitasking operating system. Multitasking means that you can have more than one application running at a time and "message based" (also referred to as "event driven") means that while an application is running, it is simply sitting in memory dormant, waiting for event messages to be sent to it by the operating system or from another running application. When you press a key on your keyboard or click your mouse, you are generating events. At the moment an event occurs, the operating system sends a message describing the event to the application that is the target of the event (i.e. the top level window for keyboard and mouse events). Messages are sent by writing them to a message "queue" for the application. When the system is idle (i.e. no programs are busy doing anything), Windows passes control to each running application in turn to give it a chance to process any event messages that it finds in its message queue. When the application is finished processing all its messages, it passes control back to the operating system so that the next running application can process its event messages. The important point here is that while one application is busy doing something, all other applications are unable to process any of their own event messages until the first application finishes.

DDE is simply a way for one application to send messages directly to another. When one program passes data or sends a DDE command to another it is simply sending a message that will get placed in the other application's message queue. The receiving application will not be able to act on that message until the sending program goes idle thus allowing it to process its messages.

If you understand what is going on behind the scenes, you can avoid some common mistakes. For example, suppose you have a device that responds to a prompt sent via the serial port. When the device receives the prompt it sends back a string of data. You also want to write an Excel macro that will send a command to the Wedge causing it to send the prompt out the serial port and then you want Excel to read the response back from the device by performing a DDERequest to the Wedge. You might come up with the following Excel macro that, although it appears to be quite logical, will never work.

```
Sub PromptMyGizmo()
  Chan =DDEInitiate("WinWedge", "COM1")          ' open a DDE link with WinWedge
  DDEExecute Chan, "[SENDOUT('?',13,10)]"        ' send a prompt out the serial port
  MyData = DDERequest(Chan, "Field(1)")          ' read back the response from Field(1)
  Sheets("Sheet1").Cells(1, 1).Formula = MyData  ' write the data to cell A1 in Sheet1
  DDETerminate Chan                              ' terminate the DDE link
End Sub
```

The reason the above subroutine will not work is because the DDEExecute statement only sends a message to the Wedge. The subroutine does not relinquish control and let the Wedge process the message until it reaches the "End Sub" statement. The DDERequest that follows the DDEExecute statement is asking for data that will not be available until the Wedge gets a chance to run; i.e. after this subroutine has finished executing.
Don't worry, there are many ways to work inside a "message based" system as you will find in the following pages of example macros and subroutines.

DDE Examples

The following pages contain example macros and subroutines for some of the more common PC application programs (i.e. Microsoft Excel, Access, Word, Visual Basic, FoxPro, Lotus 123 and Quattro Pro). The examples demonstrate how to do things such as transmit and receive serial data using Dynamic Data Exchange with the Wedge or launch and unload the Wedge from memory as well as perform various other related functions.

The examples are designed to demonstrate how to perform the most common types of serial I/O operations with the Wedge using DDE. All of the examples listed in the following pages can be found in a file called **EXAMPLES.WRI** located in your Software Wedge program directory along with all of the other Software Wedge program files. In most cases you should be able to simply cut and paste the example code from the EXAMPLES.WRI file directly into the application that you want to use. The comments in the examples will explain the details of how each routine works and will also point out any lines that may need changing in order to work with your particular system.

Important:

Dynamic Data Exchange is an extremely powerful and flexible way to pass data or commands directly from one application to another however this power and flexibility does not come without a small price in additional complexity. The examples provided below should help greatly to make the process easier. One thing to consider is that there are many situations where DDE may not be required and using the Software Wedge in "Send Keystrokes" mode may be more suitable. Using the Wedge in "Send Keystroke" mode is much easier and much quicker to configure than using the Wedge "DDE Server" mode.
For example if you are reading data from a simple instrument like a bar code reader, electronic balance or an electronic caliper and you simply want to input a small amount of data into a spreadsheet or database, then using "Send Keystrokes" mode may be a better and much simpler approach than using DDE.

There may also be situations where it might be convenient to input data from the Wedge using "Send Keystrokes" mode and also send data out the serial port by sending DDE commands to the Wedge. **The Wedge does not have to be in DDE Server mode in order to accept and process DDE commands**. You can still send DDE commands to the Wedge to have it transmit data or prompts out a serial port even when it is in "Send Keystrokes" or "Log to Disk" mode.

See Also: A Typical Software Wedge Configuration Example
The Mode Menu
Sending Keystrokes vs. Dynamic Data Exchange
Understanding Dynamic Data Exchange
Software Wedge DDE Commands

Note: TAL Technologies also maintains a website where you can find additional DDE examples along with other technical information and answers to frequently asked questions.
Our web address is: **http://www.taltech.com**

DDE Examples for Excel 5 - 2000 (Visual Basic for Applications)

Excel Example #1 - Sending DDE commands to WinWedge from a VBA subroutine

The following VBA subroutine sends a string of control codes out the serial port by issuing the DDE command “[SendOut()]” to WinWedge. This example sends an escape character (ASCII 27), a capital “P” and a carriage return-linefeed (ASCII 13 and ASCII 10).

```
Sub SendEscapeP()
ChannelNumber = Application.DDEInitiate("WinWedge", "COM1")
Application.DDEExecute ChannelNumber, "[SENDOUT(27,'P',13,10)]"
Application.DDETerminate ChannelNumber
End Sub
```

The following subroutine shows how to send a text string passed as an argument to the routine.

```
Sub SendText(StringToSend$)
ChannelNumber = Application.DDEInitiate("WinWedge", "COM2")
' The following line concatenates the SEND command with the data to be sent
Application.DDEExecute channelNumber, "[SEND(" & StringToSend$ &")]"
' The & operator is the Excel string concatenation operator thus we are concatenating
' The three strings "[SEND(" , the data in the string variable StringToSend$ and ")]"
' Thus if StringToSend$ ="ABC" then the command that is sent to WinWedge would be:
' [SEND(ABC)]"
Application.DDETerminate channelNumber
End Sub
```

The following subroutine sends a column of data out the serial port starting in the currently active cell. After each cell's data is sent, it reads the cell directly below the current cell. If the cell contains no data then it stops, otherwise it continues sending until it hits an empty cell.

```
Sub SendCells()
chan =DDEInitiate("WinWedge", "COM1")          ' open a link to winwedge on com1
MyPointer = 0                                  ' starting offset from current cell is 0
x$ = ActiveCell.Offset(rowOffset:= MyPointer, columnOffset:=0).Formula
                                               ' put string from current cell in variable x$
While Len(x$)                                  ' while the length of the string is not 0
  DDEExecute chan, "[SENDOUT(' " & x$ & " ',13)]"  ' send the string out the serial port
' the above line sends the data followed by a carriage return (ASCII 13)
     MyPointer = MyPointer + 1                 ' point to the next cell down
     x$ = ActiveCell.Offset(rowOffset:= MyPointer, columnOffset:=0).Formula   ' get next cell
Wend                                           ' loop until we reach an empty cell
DDETerminate chan
End Sub
```

The following subroutine demonstrates how to send the contents of any string variable out the serial port - even strings that contain control characters or non printable ASCII characters. The string to be sent out the serial port is passed as an argument to the SendString subroutine.

```
Sub SendString(StringToSend$)
' The following loop reads the StringToSend$ variable and generates a string containing
' the ASCII values for each character in the string separated by commas.
' This resulting string is then used as the argument to the SENDOUT command below.
' Ex: if the variable StringToSend$ = "ABC" then the variable Arg$ will be "65,66,67"
' See the syntax of the SENDOUT command in the Wedge users manual for details.
For x = 1 To Len(StringToSend$)
 Arg$ = Arg$ + LTrim$(Str$(Asc(Mid$(StringToSend$, x, 1))))
 If x <> Len(StringToSend$) Then Arg$ = Arg$ + ","
Next
channelNumber = Application.DDEInitiate("WinWedge", "COM2")
' The following line concatenates the SENDOUT command with the data to be sent
Application.DDEExecute channelNumber, "[SENDOUT(" & Arg$ &")]"
' The & operator is the Excel string concatenation operator thus we are concatenating
' The three strings "[SENDOUT(" , the data in the string variable Arg$ and ")]"
' Thus if StringToSend$="ABC" then the command that is sent to WinWedge would be:
' [SENDOUT(65,66,67)]"  - (the ASCII values for chars A, B and C are 65, 66 and 67)
Application.DDETerminate channelNumber
End Sub
```

The following subroutine demonstrates how you would call the above routine passing it the contents of a spreadsheet cell thus sending the cell contents out the serial port.

```
Sub TestSendString()
    X$ = Sheets("Sheet1").Cells(1, 1).Value
    SendString(X$)
End Sub
```

The following Excel VBA subroutine demonstrates how to transmit data out the serial port by "DDEPoking" to the Software Wedge OutputBuffer DDE item.

```
Sub PokeData()
chan = DDEInitiate("WinWedge", "COM2")          ' open a DDE channel
DDEPoke chan, "OutputBuffer", "Hello"+Chr$(13)  ' poke to the outputbuffer
' the above line pokes the word "Hello" followed by a carriage return (ASCII 13)
DDETerminate chan                               ' close the channel
End Sub
```

Excel Example #2 - Launching the Software Wedge from Excel

The following VBA subroutine executes the Software Wedge and passes it the name of a WinWedge configuration file (MyConfig.SW3) on the command line. This causes the Software Wedge to automatically load the specified configuration file and then activate itself in NORMAL mode.

```
Public Const MyPort$ = "COM1"                    ' change port if necessary
Sub RunWedge()
On Error Goto ErrorHandler                       ' Set up an error trap
AppActivate "Software Wedge - " + MyPort$        ' try to activate the Wedge Window
' the above line will generate an error if Wedge is not running or activated
On Error Goto 0                                  ' remove error trap
AppActivate Application.Caption                  ' set the focus back to excel
Exit Sub

ErrorHandler:                                    ' wedge is not running - try to launch it
RetVal = Shell("C:\WINWEDGE\WINWEDGE.EXE C:\WINWEDGE\MyConfig.SW3")
If RetVal = 0 Then                                       ' launch failed
    Beep
    MsgBox ("Cannot Find WinWedge.Exe")                  ' warn user and quit
    Exit Sub
Else                                                     ' launch succeeded
    Application.Wait Now + TimeValue("00:00:04")         ' give Wedge time to load
End If
Resume Next
End Sub
```

Note: If you name an Excel subroutine "Auto_Open()", Excel will automatically run the routine whenever the spreadsheet that contains it is initially opened. The Auto_Open subroutine is an excellent place to launch the Wedge (as well as do any other initialization functions that you might require). If you re-name the above subroutine to "Auto_Open()" then you can save yourself the step of having to manually run the above subroutine.
Similar to the Auto_Open subroutine, Excel also supports an Auto_Close subroutine that runs automatically when you close your spreadsheet. This might be a good place to tell the Wedge to quit and unload itself from memory as in the following example.

```
Sub Auto_Close()                          ' this sub runs when you close the sheet
 On Error Resume Next                     ' ignore errors
 chan = DDEInitiate("WinWedge", MyPort$)  ' open a dde link with the wedge
 DDEExecute chan, "[Appexit]"             ' tell wedge to quit
 DDETerminate chan
End Sub
```

Excel Example #3 - Requesting data from WinWedge using a VBA subroutine.

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a DDE Command Target Application, enter: **EXCEL** as the Application Name and then enter: **SYSTEM** as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input record(s) to WinWedge. When you get to the final Window with the caption “Input Record Definition Editor”, enter the string: **[RUN("GetSWData")]** as the *Field Postamble DDE Command* after the last data field that you have defined. This is a DDE command that will be sent to EXCEL after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up EXCEL:

1. Create or edit a macro module and enter the following code in the module: (To create a macro select "Tools", "Macro" and "Macros..." then enter a subroutine name and click the "Create" button.)

```
Sub GetSWData()
Static RowPointer As Long                    ' preserve variable values between calls
RowPointer = RowPointer + 1                  ' point to the next cell down
' the first time through, RowPointer will automatically initialize to point to Row 1
ChannelNum = DDEInitiate("WinWedge", "Com1") ' establish DDE link to WinWedge on Com1
F1 = DDERequest(ChannelNum, "Field(1)")      ' get Field(1) data into a variant array
WedgeData$ = F1(1)                           ' convert variant array to a string
Sheets("Sheet1").Cells(RowPointer, 1).Formula = WedgeData$
' Write the data to cell address: (RowPointer,1) in Sheet1, i.e. fill up Column 1 (Column A)
' If you are inputting multiple data fields from the Wedge you would duplicate the code above for
' each additional data field. For example, the following three lines pull in data from Field(2)
' F1 = DDERequest(ChannelNum, "Field(2)")     ' get Field(2) data into a variant array
' WedgeData$ = F1(1)                          ' convert variant array to a string
' Sheets("Sheet1").Cells(RowPointer, 2).Formula = WedgeData$ ' write the data to Column 2
DDETerminate ChannelNum                      ' kill the DDE link
' YOUR CODE GOES HERE TO DO SOMETHING WITH THE DATA MAYBE?
End Sub
```

The example above sets up the Wedge to issue a DDE command consisting of the Excel “**RUN**” command forcing Excel to run the subroutine “**GetSWData()**” after each data record is received from your serial device. The “GetSWData” subroutine performs a DDERequest to the Wedge that returns the contents of FIELD(1) to a variable named “WedgeData”. The data is then assigned to a cell in “Sheet1” and a pointer variable is updated so that the next input will be written to the cell directly below the last input. This example assumes that the name of the worksheet is **Sheet1**.

Excel Example #4 - Requesting Data from WinWedge using VBA SetLinkOnData method. (And other neat Excel tricks)

The Excel VBA macro language has a function called "SetLinkOnData" that can be used to configure Excel to automatically run a subroutine whenever data from a DDE Server is updated. In the previous example we showed how to configure the Wedge to send a command to Excel to force it to run a macro after each data record is received from a device on a serial port. This example macros show how to use the Excel SetLinkOnData method instead of having the Wedge send a command to Excel to trigger a macro to run.

The two techniques are similar in that the end result is that after each data record is received by the Wedge, Excel automatically runs a macro that grabs the data from the Wedge and does something with it. The difference is simply that in the previous example the Wedge triggers the macro to run whereas in this example Excel automatically runs the macro with no coaxing from the Wedge.

One of the benefits of the following method is that the Wedge does not have to be in "DDE Server Mode" in order to have Excel run the macro after each data record from the Wedge. Although the Wedge has an explicit "DDE Server Mode", it is actually a DDE Server no matter what mode it is in (i.e. "Send Keystrokes", "Log To Disk" or "DDE Server" modes). The reason that the Wedge has a DDE Server Mode at all is so that you can configure it to send a DDE command to another application after each input. Suppose that you wanted to log all data from the Wedge to a disk file *and* pass the data to Excel via DDE in real time. You could set up the Wedge in Log To Disk mode and use the SetLinkOnData method in Excel to trigger a macro that would pull the data from the Wedge into a spreadsheet.
Another benefit of this method is that the data from the Wedge will be transferred to Excel slightly faster than the it would if the Wedge were sending a command to Excel after each input.

In addition to demonstrating the SetLinkOnData method in Excel, the following macros also show how to launch and activate the Wedge automatically when you open up your spreadsheet as well as how to quit the Wedge automatically when you close your spreadsheet.

To use the following macros, set up the Software Wedge to work with your instrument by choosing the correct serial communications parameters and defining the input data record structure to fit the data that is returned from your serial devices. If you set up the Wedge in DDE Server mode, make sure that you do not have any "Field Postamble DDE Commands" defined. Next, activate the Wedge in Test Mode, switch to Excel and enter the following macros into a module in your Excel spreadsheet. (You may have to modify the values for some of the Global constants as described in the comments in the subroutines.) After you have entered the following subroutines, save your spreadsheet and close it. Finally, re-open your spreadsheet and start collecting data. The Wedge should automatically load and activate itself when you open the sheet.

```
Global RowPtr As Long, ColPtr As Long   ' define global variables
Global MyPort As String
Global Const CmdLine = "C:\WinWedge\WinWedge.Exe C:\WinWedge\Config.SW3"
' change the above command line to point to your copy of wedge and your config file

Sub Auto_Open()          ' this sub runs automatically when you open the spreadsheet
  FindWedge              ' check if Wedge is already running
  If MyPort = "" then                ' Wedge is not running - try to launch it
     On Error Resume Next            ' ignore errors and try to launch WinWedge
     RetVal = Shell(CmdLine)         ' launch wedge using command line defined above
     ' the above line launches and activates the wedge with a config file: Config.SW3
      If RetVal = 0 Then             ' error - wedge not found
         Beep : MsgBox ("Cannot Find WinWedge.Exe")      ' display warning
        Exit Sub                                         ' and quit
     End If                                              ' otherwise -
     Application.Wait Now + TimeValue("00:00:04")        ' give Wedge time to load
     AppActivate Application.Caption                     ' set the focus back to Excel
     FindWedge                      ' find out which port the Wedge was activated for
  End If
StartCollecting                     ' set up Excel to collect data from wedge
End Sub

Sub StartCollecting()
RowPtr = 1: ColPtr = 1              ' initialize global variables
Sheets("Sheet1").Activate           ' activate sheet 1 and set up a DDE link to WinWedge
Sheets("Sheet1").Cells(1, 50).Formula = "=WinWedge|" & MyPort & "!RecordNumber"
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!RecordNumber", _
     "GetSWData "
' the SetLinkOnData method causes Excel to run the GetSWData macro automatically when new
' data is available in the Wedge (when the RecordNumber is updated). Using the SetLinkOnData
' method in excel eliminates the need to have the Wedge send a Field Postamble DDE Command to
' Excel to cause it to run the GetSWData sub. The RecordNumber DDE item is updated after all
' other DDE items (input data fields) have been updated.
End Sub

Sub Auto_Close()         ' this macro runs automatically when you close the spreadsheet
 StopCollecting          ' set up Excel to stop collecting data from wedge
 On Error Resume Next                   ' ignore errors
 chan = DDEInitiate("WinWedge", MyPort) ' open a dde link with the wedge
 DDEExecute chan, "[Appexit]"           ' tell wedge to quit
 DDETerminate chan                      ' terminate the DDE link
End Sub
```

```
Sub StopCollecting()
Sheets("Sheet1").Activate                          ' activate sheet1
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!RecordNumber", ""
' shut down the SetLinkOnData function by assigning it an empty string to the macro name
Sheets("Sheet1").Cells(1, 50).Formula = ""         ' remove the dde link from R1C50
End Sub


Sub GetSWData()
ColPtr = 1
RowPtr = RowPtr + 1                                ' point to the next row down
' If this is the first time through then initialize RowPointer to point to Row 1 and
' ColPointer to point to Column 1 -  i.e. collect data into column 1 starting in Row 1
chan = DDEInitiate("WinWedge", MyPort)             ' initiate DDE link with wedge
MyVariantArray = DDERequest(chan, "Field(1)")      ' get Field(1) from Wedge
MyString$ = MyVariantArray(1)                      ' convert to a string
' The DDERequest function in Excel returns a variant array data type. This is a peculiarity of Excel,
' the return type should be a string data type. The above line fixes the Excel "inconsistency" and
' converts the variant array to a string by assigning element 1 of the variant array to a string
' Add your code here to do something with the data maybe?
' or possibly send a command back to the Wedge as in the following line
' DDEExecute chan, "[Beep]"  ' send a beep command to the Wedge

Sheets("Sheet1").Cells(RowPtr, ColPtr).Formula = MyString$
' the above line writes the data to cell address: (RowPtr,ColPtr) in Sheet1
DDETerminate chan                                  ' terminate the dde link
End Sub

Sub FindWedge()  ' This sub tries to find the Wedge and identify  the port that it is activated on
  MyPort = "COM1": CurrentPort = 1                 ' start with COM1
  On Error GoTo errhandler                         ' AppActivate causes an error if wedge is not active
  AppActivate "Software Wedge - " + MyPort         ' try to set focus to WinWedge
  On Error GoTo 0                                  ' disable error trap
  AppActivate Application.Caption                  ' set the focus back to Excel
  Exit Sub                                         ' quit & return to caller

errhandler:
 CurrentPort = CurrentPort + 1                     ' AppActivate generated an error - try next com port
 MyPort = "COM" + Format$(CurrentPort)             ' set MyPort to the new port (COM2, COM3, etc...)
 If CurrentPort > 4 Then                           ' if ports 1-4 have been tested and wedge not found
     MyPort = ""                                   ' set MyPort="" to indicate that wedge was not found
     Resume Next                                   ' resume at next line after AppActivate statement
 End If
 Resume                                            ' try next port - resume at AppActivate Statement
End Sub
```

Excel Example #5 - Getting past the 40 field limit in the Software Wedge

Excel 5 has two "idiosyncrasies" with its DDERequest() function. The first is that the DDERequest() function always returns a variable with the data type "Variant Array".
For some unknown reason the DDERequest function in Excel expects to receive a Tab delimited array of data. In fact, if the data from the Wedge were tab delimited, Excel would automatically parse it and fill up a Variant Array such that each tab delimited data item would be stored in successive array elements. Typically, data from most instruments is not tab delimited therefore when you perform a DDERequest to the Wedge from Excel you end up with a Variant Array variable with only a single element (element 1) containing all the data from the field in the Wedge that you are requesting the data from.

Excel will let you assign a Variant Array to a cell in a spreadsheet however if you try to perform string functions on a Variant Array, Excel generates an "Invalid Data Type" error. The Variant Array must be converted to a string data type before it can be used in a string function. This problem is easily overcome by simply assigning the Variant Array element 1 to a string. The following code fragment uses the DDERequest function to return a Variant Array and then uses an assignment statement to convert element 1 of the Array to a string.

```
DDEChannel = DDEInitiate("WinWedge", "Com2")
DDEVariantArray = Application.DDERequest(DDEChannel, "Field(1)")
DDETerminate DDEChannel
StringVariable$ = DDEVariantArray(1)          ' assign Array element 1 to a String variable
```

The second problem is that the DDERequest() function can only return string data that is less than 255 bytes unless the string is delimited with tabs (see the following section "Advanced Excel Tricks". If you try to DDERequest() a data string that is longer than 255 bytes, Excel truncates the data and you lose anything after the 255th byte.

The following subroutine shows how to get around both of the problems described above as well as how to get around the 40 field limit in the Software Wedge. To get past the 40 field limit in the Wedge, instead of using the parsing capabilities of the Wedge, we will use our own parsing routine written in Excel VBA. To do this we first have to get all the data from the device into a single string variable in Excel. This is where the problems in Excel catch up to us, especially when the total length of the data stream could be more than 255 bytes.

The following example shows how to configure the Wedge and Excel to deal with a situation where a device transmits a data record containing over 40 data fields that are delimited with commas and terminated with a carriage return. The total length of the data could also be over 255 bytes. If Excel were capable of pulling in over 255 bytes with the DDERequest function, then all we would have to do is define the "End of Record Event" in the Wedge as "Carriage Return or CrLf Received" and then define the structure of the data records as "Single Field". Finally, we would use a VBA macro to parse the string after we pass it to Excel. Because the data string could be over 255 bytes long, we have to break it into pieces before passing it to Excel and then put it back together again in our VBA macro.

The way to do this is to again select "Carriage Return or CrLf Received" as the "End of Record Event" in the Wedge and then select "Multiple Fixed Length Data Fields" as the "Record Structure". Finally, define several data fields with all their field lengths set to 250 bytes. You need to define enough 250 byte data fields so that the largest data record from the device will fit in all the fields defined in the Wedge. For example if the largest record transmitted by the device is 1125 bytes, then you would need to define five 250 byte data fields. Make sure that the Wedge is in DDE Server mode and after the very last data field that you define, enter: **[RUN("NewGetSWData")]** as the "Field Postamble DDE Command".
The above command will cause Excel to run a subroutine named "NewGetSWData" after each complete data record is received by the Wedge.
Note: Be sure not to apply any Filters to any of the data fields that you define in the Wedge.

The following subroutine performs the job of pulling the data from the Wedge and then putting it back together into one long string. Finally the subroutine parses the data into individual data fields and plugs each field into separate cells in a row of the spreadsheet.

```
Sub NewGetSWData()      ' this sub is run by the Wedge after each data record is received
Static R As Long, C As Long ' R & C point to the Row and Column where the data will go
If (R = 0) Or (C = 0) Then R = 1: C = 1                    ' make sure neither R or C is zero
Chan = DDEInitiate("WinWedge", "Com2")
' assume that there are 5 data fields defined - change to more or less if necessary
Field1 = Application.DDERequest(Chan, "Field(1)")          ' get field 1
Field2 = Application.DDERequest(Chan, "Field(2)")          ' get field 2
Field3 = Application.DDERequest(Chan, "Field(3)")          ' get field 3
Field4 = Application.DDERequest(Chan, "Field(4)")          ' get field 4
Field5 = Application.DDERequest(Chan, "Field(5)")          ' get field 5
DDETerminate Chan                                          ' kill the link

MyVar$ = Field1(1) & Field2(1) & Field3(1) & Field4(1) & Field5(1) & ","
' Convert each variant array to a string, concatenate them all together and add a final
' comma delimiter for the following parsing routine. The following code parses the string
' MyVar$ by searching for commas and placing each delimited field in a separate row
StartPos = 1                                    ' starting position in the string - start at 1st byte
While StartPos < Len(MyVar$)                    ' scan until we reach the end of the string
  DelimPos = InStr(StartPos, MyVar$, ",")       ' find the next comma delimiter
  DataPoint$ = Mid$(MyVar$, StartPos, DelimPos - StartPos)
' pull out a data point between the starting position and the position of the delimiter
  StartPos = DelimPos + 1                       ' update the starting position (skip over the delimiter)
  Sheets("Sheet1").Cells(R, C).Formula = DataPoint$ ' save the current data point
  R = R + 1                                     ' point R to the next row down for the next data point
Wend                                            ' go get the next point

R = 1               ' point R back to row 1
C = C + 1           ' increment C - move to the right one column for the next set of data
End Sub
```

Excel Example #6 - Advanced Excel Tricks - Reading large arrays of data into Excel

The native format for Excel data is tab delimited text with a carriage return at the end of each row of data. When you perform a DDERequest function in Excel, the function returns a variable with the data type "Variant Array". If the data that is requested with the DDERequest function is tab delimited with carriage returns at the end of each row of data and we assign this type of data to variant array, Excel will automatically parse the data and fill up the array with the data. If you use the FormulaArray function to assign a variant array variable to a range of cells, Excel will write the data to the specified range.

For example, if you have a device that generates data in the following format:

123,456,789,123<cr>
456,789,123,456<cr>
etc...

you can use the "pre-transfer translation table" in the Software Wedge to convert all commas to tabs so that the data would appear in the Wedge as follows:

123<tab>456<tab>789<tab>123<cr>
456<tab>789<tab>123<tab>456<cr>
etc...

Note: When the Wedge is in DDE Server mode, its default behavior is to strip out carriage returns by translating them to "Nulls" therefore we will also need to translate carriage returns back to carriage returns and not to nulls. Select "Pre Transfer Translation Table" from the DEFINE menu and select the comma character in the table and click the button marked "Translate". When the ASCII chart appears, scroll the chart down to the TAB character (ASCII 9) and click the OK button to perform the translation.
Next select the carriage return character (ASCII 13) in the translation table and click the "Translate" button again. When the ASCII chart appears this time, select ASCII 13 and click the OK button. Click the OK button in the translation table to return to the main menu.

If you define the record structure in the Software Wedge as "Single Field Data Records" and thus pull the entire array into a single field in the Wedge and then pull the array into a variant array variable using Excel's DDERequest function and finally use the FormulaArray function to assign the variant array to a range of cells, Excel will automatically parse the data and write it to a range of cells such that each cell will contain a single value from the array. Data elements separated by tabs will be written to cells across a row and each carriage return in the data causes Excel to place data following each carriage return in the next row down.

The following example demonstrates how to set up the Wedge and Excel to pull in an entire comma delimited array (where each line of data in the array is carriage return terminated). By pulling in the entire array with a single DDERequest and letting Excel parse the data for us, you can pass huge amounts of data extremely quickly into a large range of cells.

Suppose you have a device that transmits a comma delimited array of data similar to the following where the entire array is transmitted once every ten seconds or in response to a prompt and you would like to input the data into a range of cells in Excel extremely quickly: (<cr> represents carriage returns and <lf> represents linefeed characters)

123, 456, 789, 123<cr><lf>
456, 789, 123, 456<cr><lf>
789, 123, 456, 789<cr><lf>
123, 456, 789, 123<cr><lf>
456, 789, 123, 456<cr><lf>
789, 123, 456, 789<cr><lf>

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a DDE Command Destination Application, enter: “**EXCEL**” as the Application Name and then enter: “**SYSTEM**” as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input record(s) to WinWedge. Select the "Start of Record Event" as "Any Character Received" and the "End of Record Event" as "Time Delay Between Records" and then click the "Continue" button. In the next dialog box to appear prompting for the time between records, leave the default value of 3 clock ticks and click the "Continue" button again. Next, select "Single Field Data Records" from the "Record Structure" dialog box and click the "Continue" button again. When you get to the final Window with the caption “Input Record Definition Editor”, enter the string: **[RUN("GetDataArray")]** as the *Field Postamble DDE Command* for Field(1). This is a DDE command that will be sent to EXCEL after each data array is received by the Wedge.

3. Because Excel is expecting tab delimited data with carriage returns at the end of each line, we will need to use the "Pre-Transfer Translation Table" to translate the commas to tabs. Also, when the Wedge is in DDE Server mode, its default behavior is to strip out carriage returns by translating them to "Nulls" therefore we will also need to translate carriage returns back to carriage returns and not to nulls. Select "Pre Transfer Translation Table" from the DEFINE menu and select the comma character in the table and click the button marked "Translate". When the ASCII chart appears, scroll the chart down to the TAB character (ASCII 9) and click the OK button to perform the translation. Next select the carriage return character (ASCII 13) in the translation table and click the "Translate" button again. When the ASCII chart appears this time, select ASCII 13 and click the OK button. Finally, click the OK button in the translation table to return to the main menu.

4. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up EXCEL:

1. Create or edit a macro module and enter the following code in the module:
(To create a new module select “Macro” and “Module” from Excel’s “Insert” menu.)

```
Sub GetDataArray()
Static StartCol as Integer, StartRow as Integer      ' retain values between calls
' This sub performs a DDERequest for Field(1) in the Wedge and reads in a tab delimited
' array with carriage returns at the end of each line. It then fills a range of cells with the data.
' The native format for Excel data is tab delimited text with a carriage return at the end of
' each row of data.  If we assign this type of data to a range of cells using  the FormulaArray
' function, Excel automatically parses the data and fills it into the specified range.
Chan = DDEInitiate("WinWedge", "COM2")          ' initiate dde channel
MyArray = DDERequest(Chan, "Field(1)")          ' request field(1) from wedge
DDETerminate chan                               ' terminate the dde channel

' the first time through, StartCol and StartRow will be 0 - initialize StartCol and StartRow
If StartCol = 0 Then StartCol = 1   ' set the starting column where data will go in our sheet
If StartRow = 0 Then StartRow = 1' set the starting row
x = 1                               ' set default x dimension to 1
On Error Resume Next                ' ignore errors (error occurs if array has one dimension)
x = UBound(MyArray, 2)              ' get upper bound of array x dimension
On Error GoTo 0                     ' allow errors
y = UBound(MyArray, 1)              ' get upper bound of array y dimension
If x = 1 And y > 1 Then x = y:   y = 1  ' if array has one dimension then swap x and y
EndRow = StartRow + y - 1
EndCol =  StartCol + x - 1
Sheets("Sheet1").Range(Cells(StartRow, StartCol), Cells(EndRow, EndCol)).FormulaArray = _
MyArray
' fill cells in the range starting in "StartCol:StartRow" with the data array from WinWedge
' StartCol = EndCol + 1 ' un-comment this line to move right the width of the array after each input
 StartRow = EndRow + 1 ' or un-comment this line to move below previous data after each input
' keep the above 2 lines commented out to have each new array overwrite the previous one
End Sub
```

The example above sets up the Wedge to issue a DDE command to Excel forcing it to run the subroutine “**GetDataArray ()**” after each complete array is received from the serial device. The “GetDataArray” subroutine performs a DDERequest to the Wedge that returns the contents of FIELD(1) to a variant array variable named “MyArray”. The entire array is then assigned to a range of cells in “Sheet1” using the FormulaArray function. The example above assumes that the open workbook contains a worksheet named **Sheet1**.

Note: This subroutine will also work with non array type data (i.e. single data values).
Single data values are actually equivalent to a single dimension array containing a single data element. Thus you could use this example in place of Example #3.

Excel Example #7 - Continually Polling Multiple Devices In Sequence (or Sending Multiple Polls to a Single Device)

Suppose you had a group of devices connected via a multi-drop RS422 or RS485 line such that each device responded to a prompt by sending back a record of data. Suppose also that you wanted to set up the Wedge and Excel to poll each device in sequence and retrieve the data stacking up the readings from each device in separate columns in your spreadsheet. In a multi-drop situation you typically have to wait for each device to respond to its prompt before you can send the next prompt to the next device. The following instructions and macro examples show how to handle a situation like this.
Note: The following example can also be used to send multiple prompts to a single device over an RS232 connection and thus retrieve multiple data elements from the device.

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a "DDE Command Destination Application", clear out both text boxes prompting for an Application Name and a DDE topic.
Note: in this example we will not be configuring the Wedge to send a DDE Command to Excel therefore these two items are not necessary and can be left blank.

2. Select “Input Data Record Structure” in the “Define” menu and define the structure of the input data record(s) to WinWedge. When you get to the final Window with the caption “Input Record Definition Editor”, make sure that you do not have any Field Postamble DDE Commands defined.

3. Set up the Wedge to transmit the first prompt for the first device that you want to poll using either a timer controlled output string or a button controlled output string - Select "Output Strings" from the DEFINE menu. If you use a timer controlled output string, set the timer interval to a value that is high enough to guarantee that all devices will be polled before the first prompt gets sent again. Also, do not check the option "Enable Timer on Activation" - we will activate the timer later, after we are finished setting up Excel.

4. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up EXCEL:

1. Create or edit a macro module and enter the following code in the module:
(To create a new module select "Macro" and "Module" from Excel's "Insert" menu.)

```
Global Const MyPort="COM1"      ' Change port to match configuration of WinWedge
Global Const MaxPrompts% = 3                    ' number of prompts or devices
Dim RowPointer(MaxPrompts%) As Long  ' create an array of RowPointers
Dim Prompt$(MaxPrompts%)                        ' create an array of prompts

Sub GetSWData()
'  the variable PNum keeps track of which prompt was sent last  - start with device 1
Static PNum As Long ' preserve the value of PNum between calls
If PNum = 0 Then PNum = 1   ' Set the initial value of PNum to 1
' define the specific prompts for each device - add a line for each prompt
' see the Wedge manual for details about the SENDOUT command
' WinWedge will send the first prompt so we do not need to define it here
Prompt$(2) = "[SENDOUT('PString2',13,10)]"                ' prompt for device 2
Prompt$(3) = "[SENDOUT('PString3',13,10)]"                ' prompt for device 3
For x%= 1 to MaxPrompts%                                  ' initialize RowPointer array
    If RowPointer(x%) = 0 Then RowPointer(x%) = 2         ' start saving data in row 2
Next
On Error Resume Next                                      ' ignore errors
Chan = Application.DDEInitiate("WinWedge", MyPort)        ' open a link to WinWedge
MyData = Application.DDERequest(Chan, "FIELD(1)")         ' get data from field(1)
' the above line gets the response from the last prompt that was sent
MyDataString$ = MyData(1)          ' convert variant array type to a string variable
' The data from Field(1) of WinWedge is now in the variable "MyDataString$"
Sheets("Sheet1").Cells(RowPointer(PNum), PNum).Formula = MyDataString$
' the above line writes the data to column "PNum" in row "RowPointer(PNum)"
RowPointer(PNum) = RowPointer(PNum) + 1          ' increment RowPointer(PNum)
' PNum indicates which prompt requested the data that we just got
' add code here to do further processing of the data if required.
Select Case Pnum
   Case 1   ' do something with data from device #1 here if desired
   Case 2   ' do something with data from device #2 here
   Case 3   ' do something with data from device #3 here
End Select

PNum = PNum + 1          ' Increment Prompt Number - count from 1 to MaxPrompts%
If PNum > MaxPrompts% Then     ' If PNum%>MaxPrompts% then loop back to 1
 PNum = 1                      ' and quit - Wedge will send prompt for first device
Else                           ' otherwise send the prompt for the next device
  Application.DDEExecute Chan, Prompt$(PNum)  ' send the next prompt
End If
DDETerminate Chan              ' terminate the link
End Sub
```

```
Sub SetUpDDE()
Sheets("Sheet1").Activate          ' activate sheet 1 and set up a DDE link to WinWedge
Sheets("Sheet1").Cells(1, 50).Formula = "=WinWedge|" & MyPort & "!RecordNumber"
ActiveWorkbook.SetLinkOnData "WinWedge|" & MyPort & "!RecordNumber", _
     "GetSWData "

'  the SetLinkOnData method causes Excel to run the GetSWData macro automatically
'  when new data is available in the Wedge (when the RecordNumber is updated).
'  Using the SetLinkOnData method in excel eliminates the need to have the Wedge send a
'  Field Postamble DDE Command to excel to cause it to run the GetSWData sub.
'  Excel will automatically run the GetSWData sub when the Wedge updates the
'  RecordNumber DDE item. The RecordNumber DDE item is updated after each
'  data record is received.

End Sub
```

After entering the two subroutines above, run the subroutine "SetUpDDE" to establish a DDE link between Excel and the RecordNumber DDE item in the Wedge.

After Excel is set up and you are ready to start prompting for data you can enable the timer in the Wedge. What is going on here is that the Wedge will send the first prompt to the first device causing it to send back a response. When each response comes back, the Wedge updates the RecordNumber DDE item in Excel forcing it to run the macro above (because we used the SetLinkOnData method in the SetUpDDE subroutine to cause this to happen). The first thing the macro does is pull in the data from the Wedge and place it in a column using the "PNum" variable as the column selector. For example, if PNum is equal to one, the data is written to column A. If PNum is equal to two then the data is written to column B, etc.. Next, the macro increments a "PNum" variable and checks its value. If PNum is greater than the maximum number of prompts (i.e. devices) then it simply resets to 1 and quits; otherwise it transmits the next prompt in the sequence. When the response to this prompt comes in, the process repeats itself until we receive the data from the last device and PNum is incremented past the number of devices. The timer controlled output string in the Wedge starts the whole process over again by prompting for data from the first device.

Note: You do not have to use a timer controlled output string. To kick off a the sequence, simply transmit the prompt for the first device. You can do this from Excel by writing a macro that sends a DDE command to the Wedge telling it to send the first prompt or you can define a hot key or a button controlled output string in the Wedge that would transmit the first prompt.

See Also: Defining Hot Keys and Hot Key Actions
Defining Serial Output Strings

DDE Examples for Microsoft Access

Access Example #1 - Collecting Data Directly To A Table In An Access Database

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the “Mode” menu in the Wedge. When the dialog box appears asking for a DDE Command Destination Application, enter: “**MSACCESS**” as the Application Name and the name of your open database (without the MDB Extension) as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input data to WinWedge. When you get to the final Window entitled “Input Record Definition Editor”, enter the string: **[GrabData]** as the Field Postamble DDE Command after the last data field that you have defined. This is a DDE command that will be sent to MSACCESS after each data record is received by the Wedge.

3. Set up the rest of the Wedge as needed for your serial device and then activate it.

Steps for setting up MS ACCESS:

MSAccess does not have a “GrabData” DDE command so the first step is to create one. When you define a macro in an Access database, the name of the macro automatically becomes a valid MSAccess DDE command for as long as the database is open.

1. Create a new macro that performs the “RunCode” Action. For the function name that is required by the RunCode action, enter the name “**GetWedgeData()**”. Save the macro with the macro name “**GrabData**” and finally, create a new module with a function named “GetWedgeData()” and enter the following commands in the body of the function:

```
Function GetWedgeData ()
Dim ChannelNumber, MyData As String                 ‘create variables
ChannelNumber = DDEInitiate("WinWedge", "Com1")     ‘open a link to WinWedge
MyData = DDERequest(ChannelNumber, "FIELD(1)")      ‘get data from field(1)
DDETerminate ChannelNumber                          ‘ terminate the link
If Len(MyData) = 0 Then Exit Function               ' if no data then quit
' Data from Field(1) in WinWedge is now in the variable “MyData”. The following code adds
' a new record to TABLE1 and stores the data in a field named "SerialData"
Dim MyDB, MyTable                   ' dim variables for database and recordset
Set MyDB = DBEngine.Workspaces(0).Databases(0)
Set MyTable = MyDB.OpenRecordset("TABLE1", DB_OPEN_TABLE)
MyTable.AddNew          ' add a new record - the new record becomes the current record
MyTable("SerialData") = MyData          ' write our data to a field named "SerialData"
MyTable.Update                          ' update the table
MyTable.Close                           ' close the table
End Function
```

The example above sets up the Wedge to issue a DDE command consisting of the name of an Access macro (directly to the Access database that contains the macro) after each data record is received from your serial device. The macro calls an AccessBASIC function that performs a DDERequest back to the Wedge and returns the contents of FIELD(1) to a variable named "MyData". It then creates a new record in a table and inserts the data into the new record.

Access Example #2 - Launching And Terminating The Software Wedge From Access

```
Function LaunchWedge ()
' This function will launch the Wedge feeding it the name of a configuration file on the
' command line causing the Wedge to automatically load the config file and activate itself
CmdLine = "C:\winwedge\winwedge.exe C:\winwedge\test.SW3"
' Change "CmdLine" to specify the correct path for your copy of WinWedge.Exe
' Make sure that the complete path is specified for your configuration file as well
RetVal = Shell(CmdLine)
x = Now + TimeValue("00:00:03") ' wait for roughly 3 seconds
Do While Now < x                ' give wedge time to load and activate itself
    DoEvents                    ' allow other Windows processes to continue
Loop
If RetVal = 0 Then              ' launch failed
  Beep : MsgBox ("Cannot Find WinWedge.Exe")  ' warn user and exit
  Exit Function
Else                                          ' launch succeeded
    AppActivate "Microsoft Access"            ' set the focus back to Access
End If
End Function

Function KillWedge ()           ' This function unloads the Wedge from memory
chan = DDEInitiate("WinWedge", "Com1")' initiate DDE channel with wedge on COM1
DDEExecute chan, "[AppExit]"          ' send AppExit command to Wedge
DDETerminate Chan                     ' terminate the DDE link
End Function
```

Access Example #3 - Transmitting Variables Out The Serial Port From Access

```
Function SendString (StringVar$)
' The Wedge must be running and activated for COM1 for this code to work
chan = DDEInitiate("WinWedge", "Com1")' initiate DDE channel with wedge
DDEExecute chan, "[SEND(" + StringVar$ + ")]"   ' send String out the serial port
DDETerminate chan                               ' terminate the link
End Function
```

For more examples of how to transmit data, refer to the examples for Microsoft Excel VBA on page 84. Excel VBA and AccessBasic are almost identical macro languages.

DDE Examples for Microsoft Word for Windows

Steps For Setting Up The Software Wedge:

1. Select “DDE Server” from the MODE menu and enter: **WinWord** as the Application Name and **System** as the DDE topic in the dialog box that appears and click the OK button.

2. Select “Input Data Record Structure” from the DEFINE menu and define the structure of the input records to the Wedge. When you get to the final Window entitled “Input Record Definition Editor”, enter the string: **[GRABDATA()]** as the "Field Postamble DDE Command" after the last data field that you have defined. This is a DDE command that will be sent to Word after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and activate it.

Steps For Setting Up Word:

When you create a WordBasic macro or subroutine in Word and save it in the Global macro sheet, the name of the subroutine automatically becomes a valid Word DDE command. Create a new macro by selecting "Macro" from the TOOLS menu. When the dialog box appears prompting for a macro name, enter the name: GRABDATA and then click the button labeled "Create". Enter the following code in the macro editor window:

```
Sub GrabData()
chan = DDEInitiate("WinWedge", "COM2")          ' open a link to the Wedge
MyData$ = DDERequest$(chan, "Field(1)")         ' get the data from the Wedge
DDETerminate chan                               ' close the link
WordBasic.Insert MyData$                        ' enter the data into the open document
WordBasic.InsertPara                            ' insert a paragraph marker (hard return)
' add your own code here
End Sub
```

After you are done entering the code select "Close" from the FILE menu to save the macro.
This example sets up the Wedge to issue a DDE command (**[GRABDATA()]**) to Word after each data record is received from your serial device. This causes the **GRABDATA** subroutine to run which then performs a DDERequest back to the Wedge asking for the data in Field(1) in the Wedge Window. This data is returned to Word in the variable: MyData$. Once the data from the Wedge is in a variable, you can do whatever you like with it by including additional code of your own design. This example simply inserts the text in the MyData$ string variable into the open document and then inserts a paragraph marker to move the cursor to the next line down.

The following code fragment shows how to transmit a string out the serial port from Word:

```
channel = DDEInitiate("WinWedge", "COM2")       ' open a link to the Wedge
DDEExecute channel, "[Sendout('Test',13)]"      ' transmit "Test" and a carriage return
DDETerminate channel                            ' close the link
```

DDE Examples for Visual FoxPro for Windows

FoxPro Example #1 - Collecting data into MS FoxPro using the Software Wedge.

Steps For Setting Up The Software Wedge:

1. Select “DDE Server” from the Software Wedge MODE menu. When the dialog box appears asking for a DDE Command Destination Application, enter: **MyServer** as the Application Name and enter **MyTopic** as the DDE topic.

2. Select “Input Data Record Structure” from the DEFINE menu and define the structure of the input records to WinWedge. When you get to the final Window entitled “Input Record Definition Editor”, enter the string: **[GRABDATA]** as the "Field Postamble DDE Command" after the last data field that you have defined. This is a DDE command that will be sent to your FoxPro application after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and activate it.

Steps For Setting Up FoxPro:

FoxPro does not have a “GrabData” command so we must create one. The FoxPro code example on the following page does exactly that. Create a new program module in the CODE section of FoxPro and enter the code on the following page. After you are done entering the code save your work and run the program.

What this example does is to set up the Software Wedge to issue a DDE command **[GRABDATA]** to your FoxPro application after each data record is received from your serial device. This causes the procedure **DoCommand** to run which then performs a DDERequest back to the Wedge asking for the data in Field(1) in the Wedge Window. This data is then returned to FoxPro in the variable: MyData. Once the data from the Wedge is in a variable, you can do whatever you like with it by including additional code of your own design.

```
*** Set this application up as a DDE server. The following code must be run at the start***
*** of your application to make it able to respond to DDE commands.***
*** Set the DDE application name to: "MyServer" and Enable DDE command execution***

= DDESetService('MyServer', 'DEFINE')
= DDESetService('MyServer', 'EXECUTE', .T.)
*** Set the DDE topic to "MyTopic" ***
*** Set up procedure "DoCommand" to run on all DDEExecutes to topic "MyTopic"***
= DDESetTopic('MyServer', 'MyTopic', 'DoCommand')



*** The following procedure will run whenever another application***
*** issues a DDE command to application name 'MyServer' and topic 'MyTopic'***
*** The DDE command is passed in the variable "gData"***

PROCEDURE DoCommand
PARAMETERS gnChannel, gcAction, gcItem, gData, gcFormat, gnAdvise
glResult = .F.
*** It's necessary to return .T. from an INITIATE action or no connection is made ***
IF gcAction = 'INITIATE'
     glResult = .T.
ENDIF
IF gcAction = 'EXECUTE'

*** support a GRABDATA command - performs a DDERequest to WinWedge***
*** on COM2, requests data from FIELD(1) and stores it in a variable: "MyData"***

     IF gData="[GRABDATA]"
        gnChanNum = DDEInitiate('WinWedge', 'COM2')
        IF gnChanNum != -1
           MyData=DDERequest(gnChanNum,'Field(1)')
           = DDETerminate(gnChanNum)      && Close the channel
*** The variable 'MyData' now contains the data from Field(1) in WinWedge***
*** Your code goes here to do something with it ***
           WAIT WINDOW MyData NOWAIT  && For Example - Display the data
        ENDIF
     ENDIF
ENDIF

IF gcAction = 'TERMINATE'
     glResult = .T.
ENDIF
RETURN glResult
```

FoxPro Example #2 - Transmitting Data Out The Serial Port From FoxPro

*** the following procedure demonstrates how to transmit a string out the serial port
*** by sending a DDE command from FoxPro to WinWedge

```
PROCEDURE TransmitTesting

CmdString="[SENDOUT('testing 123',13)]"
*** transmit the string "testing 123" followed by a carriage return (ascii 13)
gnChanNum = DDEInitiate('WinWedge', 'COM2')
IF gnChanNum != -1
 glExecute = DDEExecute(gnChanNum, CmdString)
 = DDETerminate(gnChanNum)     && Close the channel
ENDIF
RETURN
```

*** the following procedure is a variation of the above procedure that transmits a
*** string passed as a parameter to the procedure (gData)

```
PROCEDURE TransmitData
PARAMETERS gData
cmdstring="[SENDOUT(' " & gData & " ' )]"
*** transmit the string passed in the variable gData
*** note the use of both single and double quotes around the gData variable
*** refer to the syntax for the SENDOUT command in the Wedge manual for details
gnChanNum = DDEInitiate('WinWedge', 'COM2')
IF gnChanNum != -1
 glExecute = DDEExecute(gnChanNum, CmdString)

 = DDETerminate(gnChanNum)     && Close the channel
ENDIF
RETURN
```

DDE Examples for Microsoft Visual Basic for Windows

Linking a VB Text Box to a Software Wedge Data Field

To link a data field in the Wedge to a text box in a VB application you simply set the properties for the text box to specify a DDE link. For example the following code would establish a "Hot" link between field(1) in the Wedge (active on COM2) so that whenever data changes in Field(1) of the Wedge, it would also automatically change in the text box Text1. When a data field from the Wedge is linked to a text box on a VB form using a "Hot" link, the text box will also receive a "Change" event whenever the data from the Wedge changes. If you set the LinkMode property to "Notify", the text box will receive a "LinkNotify" event whenever the Wedge updates the data for Field(1) - even if the data does not change. Note: If you set the LinkMode to "Notify" your Text1_LinkNotify event handler will have to perform a "LinkRequest" method to update the textbox. (i.e. Text1.LinkRequest)

```
Hot = 1 : Warm = 2 : Notify=3 : None = 0
Text1.LinkTopic = "winwedge|com2"          ' set the link topic
Text1.LinkItem = "field(1)"                ' set the link item
Text1.LinkMode = Hot                       ' set the link mode to "Hot"
```

Sending DDE Commands To The Software Wedge From A VB Application.

After a link is established between a text box and a data field in the wedge, you could issue commands to the Wedge using the text box "LinkExecute Method". For Example the following code sends the string "Testing" followed by a carriage return out the serial port.
Note: The Wedge must be active on the specified serial port for the following code to work.

```
Hot = 1 : Warm = 2 : Notify=3 : None = 0
Text2.linkmode = None                      ' disable links before setting topic
Text2.LinkTopic = "winwedge|com2"          ' set the link topic
' there is no need to set the LinkItem property when sending DDE commands
Text2.LinkMode = Warm                      ' set the link mode to "Warm"
Text2.LinkExecute "[Sendout('Testing',13)]"  ' send the string out the serial port
```

Launching The Software Wedge From A Visual Basic Application

```
Sub RunWedge()
' the shell statement passes a wedge config file as a command line arg.
' This causes the wedge to automatically load the config file and activate itself.
x% = Shell("c:\winwedge\winwedge.exe c:\winwedge\ddetest.SW3")
ThreeSecsFromNow = TimeValue(Now) + TimeValue("00:00:03")
Do While TimeValue(Now) < ThreeSecsFromNow
 DoEvents                          ' give wedge time to load
Loop
End Sub                            ' WinWedge should be loaded now
```

DDE Example for Lotus 123 for Windows

The following is an example of two Lotus 123 macros that work together to take data readings from a DDE conversation with the Software Wedge and place them in a column such that each new reading is added to the bottom of the column.

Start_Wedge **{LET A1,1}**
{LAUNCH "C:\WinWedge\WinWedge.Exe C:\WinWedge\MyConfig.SW3"}
{DDE-OPEN "WinWedge","COM1"}
{DDE-ADVISE "Get_Data","RecordNumber"}
{DDE-EXECUTE "[BEEP]"}

Get_Data **{DDE-REQUEST ("B"&@STRING(A1,0)),"Field(1)"}**
{LET A1,A1+1}

The macro named "Start_Wedge" first sets up a counter value (equal to one) in cell A1 and then launches the Software Wedge so that it will automatically load and activate itself using the configuration file named "MyConfig.SW3". Next, it opens a DDE conversation between Lotus and the Software Wedge using the DDE-OPEN macro command. The following DDE-ADVISE statement sets up Lotus to run the second macro "Get_Data" each time the data in the RecordNumber DDE item in the Software Wedge changes. The last line in the "Start_Wedge" macro sends the DDE command "[BEEP]" to the Software Wedge and is included here as an example of how to issue DDE commands to WinWedge from Lotus.

If the Start_Wedge macro is successful, then the second macro named "Get_Data" will run automatically whenever the RecordNumber DDE item in the Software Wedge is incremented (i.e. after each new data record is received). The DDE-REQUEST command in the Get_Data macro simply retrieves the data from Field(1) in the Software Wedge and places the data in column B in the row pointed to by the value in cell A1. The line {LET A1, A1+1} increments the value in cell A1 so that the next time through, the data from Field(1) will be placed in the cell directly below the last data value.

To use this example, you would simply run the macro "Start_Wedge" one time when you want to start collecting data. This macro assumes that the Software Wedge will be activated for serial port COM1.

DDE Examples for Quattro Pro for Windows

Steps for setting up the Software Wedge:

1. Select “DDE Server” from the Software Wedge “Mode” menu. When the dialog box appears asking for a DDE Command Destination Application, enter: “**QPW**” as the Application Name and then enter “**SYSTEM**” as the DDE topic.

2. Select “Input Record Structure” in the “Define” menu and define the structure of the input record(s) to WinWedge. When you get to the final Window entitled “Input Record Definition Editor”, enter the string: **{Get_Wedge_Data}** as the Field Postamble DDE Command after the last data field that you have defined. This is a DDE command that will be sent to Quattro Pro after each data record is received by the Wedge.

3. Set up the rest of the Software Wedge parameters and then activate it.

Steps for setting up Quattro Pro:

Quattro Pro does not have a “Get_Wedge_Data” DDE command so the first step is to create one. When you create and name a macro in a Quattro Pro notebook, the name of the macro automatically becomes a valid Quattro Pro DDE command for as long as the notebook is open.

1. Enter the following lines into cells a Quattro Pro notebook and name each cell or group of cells using the block names indicated below.

Block Name	**Block Contents**
row	**1**
column	**A**
cell_number	**A1**
dde_channel	**0**
dde_command	**[beep]**
result	**0**
get_wedge_data	**{ONERROR _error_handler}** **{REQUEST dde_channel,"FIELD(1)",@@("cell_number")}** **{LET row,row+1}** **{LET cell_number,+column&@string(row,0)}** **{EXECUTE dde_channel,+dde_command,result}**
_error_handler	**{INITIATE "WinWedge","COM1",dde_channel}**

This example sets up the Software Wedge to issue a DDE command to Quattro Pro consisting of the name of a macro (get_wedge_data) after each data record is received from your serial device. The macro performs a DDE Request to the Software Wedge that returns the contents of FIELD(1) to a spreadsheet cell pointed to by the variables named “row” and “column”. (i.e. row 1, column A)

The blocks with the names “row”, “column”, “cell_number”, “dde_channel”, “dde_command” and “result” are simply variables that are used by the “get_wedge_data” macro. The “get_wedge_data” macro first sets up an error handler pointing to the macro named “_error_handler”. The following line then attempts to request the data for “Field(1)” from the Software Wedge and store the data in a cell whose address is pointed to by the variable “cell_number”. The first time the macro is invoked the “request” command will fail because a DDE channel has not yet been opened. The purpose of the “_error_handler” macro is to simply initiate the DDE channel when the first call to “get_wedge_data” fails. After the “request” macro command, the variable “row” is incremented and the variable “cell_number” is updated using the new row number so that it points to the cell directly below the one where the last data reading from the Software Wedge was stored. Thus, each time a new data record is received by the Software Wedge, the contents of “Field(1)” will be retrieved and stacked up in a column. The final macro line in the “get_wedge_data” macro uses Quattro Pro’s “Execute” command to issue a “Beep” command to the Software Wedge. This line has been included as an example of how to send DDE commands to WinWedge from Quattro Pro.

Passing Data Across a Network Using Network DDE

Windows supports a feature called "Network DDE" that allows DDE data to be passed across a network from one PC to another. This means that it is possible to have the Software Wedge running on one or more PCs providing real time data to an application running on an entirely different PC in the network. Note: Network DDE must be enabled in Windows 95/98, NT or 2000 by running the program NETDDE.EXE. You may want to place the NETDDE.EXE program in your StartUp folder so that it runs automatically when you start Windows 95/98, NT or 2000.

The process of setting up a network DDE connection is fairly simple and involves using the ClipBook application found in the MAIN program group in Windows. (In Windows 9x, the ClipBook is not normally get installed however it can be found on your Windows 9x setup disks or CDROM.) The ClipBook extends the concept of the Windows Clipboard across a network. The ClipBook allows you to create "ClipBook Pages" that can be "Shared" with other computers on a network. Once you create and share a page on one computer, other computers on the network can "connect" to that page. If the page contains data from a DDE server (like the Wedge), then you can set up DDE links to the data and the data will be passed to any client application on the network just as if the client were running in the same local PC as the server application.

Suppose you have the Wedge running in one PC and you want to DDE link data from "Field(1)" in the Wedge to a cell in an Excel spreadsheet running in another PC on your network. The following steps outline the procedure:
Note: All versions of Excel prior to 5.0c have a bug that causes Network DDE connections to fail. If you have an earlier copy of Excel you can get an update from Microsoft for free.

1. Load and activate the Wedge on the first PC. When the Wedge window appears, place the cursor in input data field "Field(1)" and select "Copy" from the EDIT menu.

2. Open the ClipBook program in the PC where the Wedge is running and select "Local ClipBook" from the WINDOW menu and then select "Paste" from the EDIT menu. In the dialog box that appears, type **MyPage** in the text box labeled "Page Name" and make sure that the check box labeled "Share Item Now" is checked.

3. Open up the ClipBook program on the PC where Excel will be run and select "Connect..." from the FILE menu. When the dialog box appears, enter the network name of the PC where the Wedge is running and click the OK button. (If you do not know the name of the computer, click the "Browse" button and the ClipBook will display a list of all computers in the Network). This will display a new window that will contain all the "shared" items in the ClipBook running on the other PC. You should see an item in the window with the name "MyPage" and then select "Copy" from the EDIT menu. This operation takes the data from the ClipBook on the other PC and places it in the local clipboard.

4. Open Excel and place the cursor in the cell that you want to link the data to and select "Paste Special" from Excel's EDIT menu. In the dialog box that appears, check the option labeled "Paste Link" and then click the OK button. If the link is successful, you will see the data from Field(1) in the cell and you will also see the formula for the link in the formula bar. Whenever new data is received by the Wedge it will be updated in the spreadsheet cell.

If you look at the formula that Excel returns from the above Copy / Paste Link procedure, it will look something like the following:

=\\ComputerName\NDDE$|$MyPage.dde!Field(1)

The formula for a DDE link in Excel is expressed using the DDE Application Name followed by a pipe character (|), a DDE Topic followed by an exclamation mark and then the DDE Item Name.

If you were to use the Copy / Paste Link method of establishing a DDE link between the Wedge and Excel where both programs were running on the same PC, the linked cell in Excel would contain a formula similar to the following:

=WinWedge|COM1!Field(1)

The difference between a local DDE connection and a network DDE connection is that instead of using the DDE Server's executable file name, the Network DDE Application Name becomes two slashes followed by the name of the computer running the DDE Server program followed by a single slash and the word: **NDDE$**. The DDE Topic also changes to a dollar sign followed by the "Page Name" that was assigned in the ClipBook application followed by a period and the word **dde**. The DDE Item Name stays the same.

After you create a "Page Name" using the ClipBook application and paste data from a DDE server into the page, the page name and the DDE connection data contained in it will continue to exist even after you close the ClipBook application on both computers. You can even shut down Windows on both computers and when you restart you will be able to re-establish the DDE links using the same Network DDE Application, Topic and Item Names.
In other words, after you set up a Network DDE connection once, you do not need to go through the same procedure in the ClipBook every time you want to re-establish the link.

If you wanted to send a DDE command from Excel to a copy of the Wedge running on a remote PC, all you need to do is to change the DDE Application Name and DDE Topic to the new Network DDE Application and Topic Names. For example the following Excel macro could be used to send the "BEEP" command to a Wedge on a remote PC. (We will assume that the name of the remote PC is "Bob" and that a ClipBook page was created on that PC using the name "MyPage" and that the page contains a DDE data field from the Wedge.)

```
Sub BeepRemoteWedge()
channelNumber = Application.DDEInitiate("\\BOB\NDDE$", "$MyPage.DDE")
Application.DDEExecute channelNumber, "[BEEP]"
Application.DDETerminate channelNumber
End Sub
```

Diagnosing Serial Communications Problems

To diagnose communications problems go throught the following steps:

Step1: Select **Settings** from the **Port** menu in the Wedge and choose the serial communications parameters required for your serial device.
Note: All serial communications parameters should exactly match the settings for your serial device. If you do not know the correct parameters, then you will have to contact the manufacturer of the device to obtain this information.

Step 2: Open the Port Analyze window by selecting **Analyze** from the **Port** menu. If you get an error message that reads "Device COMn Not Available" then either the specified serial port is currently being used by another program (possibly another instance of the Wedge), the port does not exist, or the Windows COM Driver for the port has not been installed or set up correctly in the "PORTS" section of the Control Panel. If you suspect that the port may be set up incorrectly in the control panel, then refer to your Windows User's Manual and the help files available in the Control Panel "PORTS" dialog box for information on how to correct or work around the problem.

Step 3: If the Port Analyze window appears, transmit some data from your serial device to the Wedge. The data from the device should appear in the text box marked "Input Buffer". If no data appears in the input buffer then either your instrument is not connected to your PC using a proper serial cable; you are not connected to the serial adapter that you specified in the port settings dialog box or your device is not transmitting any data.

Try plugging the cable from the device into a different serial port and transmitting data again to rule out the possibility that you are plugged into the wrong serial port.

If you are certain that the device is transmitting data and yet you still have no success receiving data in the Analyze Window, then you may need a 'NUL MODEM ADAPTER' or a different serial cable. If you normally connect your device to a serial printer and are able to print successfully, then you definitely need a nul modem adapter. A nul modem adapter is a small plug that connects between a serial cable and your computer that simply crosses the transmit and receive lines in the cable so that the transmit line from your serial device is connected to the receive line on your PC and vice versa. Nul modem adapters are available at most computer supply stores for under five dollars.

If your PC beeps one or more times whenever you try to read data into the Input Buffer in the Port Analyze Window or if data appears in the Input Buffer but part or all of it is either garbled or consists of unreadable characters, then you are connected to the correct port however the communications parameters that you chose in the Port Settings dialog box are wrong and must be corrected.

Note: The Wedge has an option "Beep On Input" (located in the "Port" menu) that configures the Wedge to beep your PC's speaker whenever it receives a data record through the serial port. The "Beep On Input" option does not apply to data received in the Port - Analyze window. If your PC beeps while inputting data into the Port Analyze window then it is an indication of a serial communications error; typically either a mismatch in serial communications parameters (baud rate, parity, databits, etc.) between the Wedge and your device or a serial buffer overrun.

Troubleshooting

If you are not receiving any data all in the Port Analyze Window:

1. Make sure that the PC you are using is equipped with the serial port selected in the Port Settings Dialog box. If you do not have the chosen serial port or if your serial port is configured improperly then you must correct the problem and try again.

Important: Make sure that the Port Address and IRQ number in the Advanced section of the PORTS dialog box in the Windows Control Panel are correctly specified for your serial adapter hardware. The standard values for COM1 through COM4 are shown below:

Connector	Port Address	IRQ
COM1	03F8	4
COM2	02F8	3
COM3	03E8	4
COM4	02E8	3

See Also: Running the Wedge On Multiple Serial Ports Simultaneously

2. Check that you are connected to your PC using the proper RS232 cable for connection to a PC and that you are plugged into the serial adapter that you specified in the Port Settings Dialog box. Also make sure that your serial device is getting power and is turned on. You may also want to contact the manufacturer of the device to confirm that you are using the correct cable and that you are doing everything that is necessary to connect to a PC and also to get the device to transmit data.

3. Use the Port Analyze window in the Software Wedge and try, at least, to get some data from the device to appear in the Input Buffer text box. If you cannot get any data at all to appear (even garbage) then either the device is not transmitting anything, or you need a "NUL Modem Adapter". A nul modem adapter is a small plug that connects between the serial cable from your device and serial port on your PC. Its purpose is essentially to cross the transmit and receive lines in the cable so that the transmit line from your serial device is connected to the receive line on your PC and vice versa. If you normally connect your serial device to a printer and you can successfully print to the printer, then you definitely need a NUL Modem Adapter. You can pick one up at any Radio Shack or computer supply store for about five dollars. Take the device or the cable to the store with you so that you can match the adapter to the cable.

4. The Analyze Window in the Wedge is similar to the Terminal program that comes with Windows 3.1 or the HyperTerminal program that comes with Windows 95/98, NT or 2000 (located in your Accessories program group). The manufacturer of your serial device may not be familiar with the Software Wedge however they will more than likely be familiar with either Terminal or HyperTerminal. A good course of action at this point would be to contact the manufacturer of your serial device and have one of their technical support representatives walk you through the process of connecting the device to the Terminal or HyperTerminal program. Once you are able to get data into Terminal or HyperTerminal you should have no trouble doing the same thing with the Wedge.

If data appears garbled or unreadable in the Port Analyze window:

5. Check that the serial device is set up using exactly the same communications parameters as the Wedge, i.e. Baud rate, Parity, number of data bits, and number of stop bits. If you do not know the parameters used by your device, then you will need to either consult the users manual for the device or contact its manufacturer for this information. You can also use the Port Analyze feature in the Software Wedge to try different parameters until you get data that appears correct. Try different baud rates first using No Parity, Eight Data Bits, and One Stop Bit until you get data that looks either partially or completely correct. Finally, try different combinations of Parity, Number of Data Bits and Number of Stop Bits until all data is correct. Most devices use either seven data bits with Even or Odd parity or eight data bits with No Parity. One stop bit is also used more frequently than two.

If data appears using the Port Analyze feature but not after you activate the Wedge:

6. Make sure that the Wedge is activated and enabled and try reading data into the NotePad application that is shipped with Windows. If no data appears in NotePad, then you most likely have the Wedge configured incorrectly (either the Start or End Of Record Events are wrong or your parsing and filtering parameters are specified incorrectly). If you are inputting very large data records, you may also need to increase the size of the serial input buffer. Re-configure the Wedge and try again.

If your computer locks up when you open the Analyze window or activate the Wedge:

Microsoft has confirmed that there is a “small bug” in the Windows 3.1 COM driver SERIAL.386 that causes a system crash when you try to activate a serial port. The bug only occurs when running an older copy of Windows for Workgroups 3.11 on a fast 486 or Pentium PC with 16550 UARTs (high speed serial adapters). The bug has been fixed and a replacement SERIAL.386 driver file has been made available. The new driver can be found on the Software Wedge distribution diskette in a self extracting ZIP file named WG1001.EXE. To extract the new driver, copy the WG1001.EXE file to your hard drive and then run it to un-compress the files in it. When you un-compress the WG1001.EXE file, a README file will also be un-compressed along with the new SERIAL.386 driver. This README file contains complete instructions for replacing the defective driver as well as some technical notes from Microsoft regarding the symptoms of the bug.

If you cannot get the Wedge to operate properly after you have gone through the above procedures, then call or fax TAL Technologies for assistance at: Tel: (215)-763-5096 or Fax:(215)-763-9711 during regular business hours. Please have this manual at hand and be at your PC with the Wedge running and your serial device connected when you call for support. A valid serial number is also required to obtain telephone support. Your serial number is located either on the back cover of your users manual or on a sticker on the outside of the box that the Software Wedge was shipped in.

TAL Technologies maintains a web site at **http://www.taltech.com** where you can find the latest technical notes and articles relating to typical support issues for the Software Wedge. The web site also contains many configuration examples for the Wedge as well as sample Excel spreadsheets that show how to perform the most common functions with the Wedge. Web based support is available 24 hours a day and you may also email support questions to: **support@taltech.com**.

Running The Wedge On Multiple Serial Ports Simultaneously

The Software Wedge may be activated for multiple serial ports simultaneously provided you have the proper serial adapter hardware. To do so, simply create two configuration files (one for each port) and launch the Wedge twice from the Program Manager and activate each *instance* of the Wedge using the two configurations. If you have just COM1 & COM2 then you should have no problems using the Wedge with both ports simultaneously.

Using More Than Two Serial Adapters At A Time

If you will be using the Wedge on a PC that is not a Micro Channel or EISA Bus PC and more than two serial ports will be active at the same time, then you may have to change the configuration of your serial adapters for ports COM3 and above.

The standard bus architecture (ISA bus) for all IBM or compatible 80x86 PC's provide dedicated interrupt request lines for only two serial adapters (COM1 uses IRQ4 & COM2 uses IRQ3). Many serial adapters that provide COM3 & COM4 also use the same IRQ lines as COM1 & COM2. When interrupt driven serial I/O is used, as by the Wedge, then two active serial ports will end up fighting with each other for control of the shared interrupt request line and one or both will lose. One adapter will try to drive the IRQ line "high" while the other adapter tries to drive it "low". EISA and Micro Channel bus PC's avoid this problem by using a level triggered interrupt control mechanism instead of the original level triggered scheme used in the ISA bus.

One way around this problem is to take advantage of an unused IRQ line on your PC (usually IRQ 2, 5, 7,10,11,12 or 15) by configuring your COM3 or above adapter's jumper or switch settings to use one of these IRQ lines instead of IRQ4 or IRQ3. Most add-on serial adapters provide jumpers or switches just for this purpose. This must be done with care because many other peripheral devices including mice, network cards, internal modems and sound cards may also use these IRQ lines. You can check to see which IRQ lines are free by using the Microsoft Diagnostics program that comes with DOS. Exit Windows and type: **MSD** at the DOS prompt and when the menu appears, select "IRQ Status". IRQ lines that are available for use will be specified as "Reserved".

Note: If you change the IRQ for a serial adapter, you also have to inform Windows of the change by modifying the Interrupt Request Line value for the port in the Windows Control Panel. (Open the Control Panel from the MAIN program group, double click on the PORTS icon then select the COM port that you are changing the IRQ for. Next, click on the SETTINGS button and then click the ADVANCED button in the Settings dialog box. Finally, choose the new IRQ for the COM port in the Advanced Settings dialog box).

Another solution is to use a multi-port serial adapter that provides an "interrupt multiplexing" scheme. Digi International (612)-943-9020 carries a complete line of multi-port serial adapters complete with replacement Windows COM drivers that will allow you to use all ports simultaneously. DigiBoard carries a wide range or serial I/O boards complete with Windows COM drivers and they also provide excellent technical support for their products.

Understanding The Data From Your Serial Device

Before you can correctly configure the Software Wedge to parse and filter data received from your serial device, you must first have a complete understanding of the data that your device transmits. You should also be able to recognize what parts of the data are important to you and also completely understand what features of the Software Wedge can be used to manipulate your data. To use the Software Wedge effectively, you must think of each input from your device as a *record* of information containing one or more specific *fields* of data. The next step is to decide what features of the Wedge can be used to consistently separate each data record from the next and also what features can be used to separate or *parse* individual data fields within a record from the next field. Finally you need to identify which fields within each record are important to your application and which fields should be removed or ignored.

The best place to start is to refer to the users manual for the device that you are using. The users manual should have a section that describes the structure and contents of all data that can be outputted from the device. If you do not have a users manual for your device or if the output data structure is not described, you can use the Port-Analyze dialog box in the Software Wedge or the Terminal program that is provided with Windows to manually analyze the output from your device by actually viewing the data that is transmitted from it.

When you configure the Software Wedge for a particular device you must define the input record structure for your serial data to the Wedge by selecting "Input Data Record Structure" from the *Define* menu. When defining the input record structure, you specify a "Start Of Record Event", an "End Of Record Event" and a general record structure for each data record (i.e. single field records, multiple delimited fields, or multiple fixed length fields).

With some devices, the record structure may be immediately obvious. For example, the following data record contains three numeric data fields delimited by commas and terminated by a carriage return-linefeed pair:

110,250,801<CrLf>

For this type of data record, you might select "Any Character Received" as the *Start Of Record Event* and "Carriage Return or CrLf Received" as the *End Of Record Event* and for the *Record Structure* you would choose "Multiple Delimited Data Fields" and specify that there are three data fields per record with a comma delimiter separating each field.

Note: The <CrLf> above refers to a carriage return and a linefeed character. A carriage return (Cr) will appear in the "Analyze" window in the Wedge as a musical note and a linefeed (Lf) will appear as a small rectangle with a circle inside it.

Other devices may output data with a record structure that is less obvious than the example above. Consider the following two possible (different) output records transmitted from one particular laboratory instrument:

Sample#1,*213**32****23**<CrLf>
Sample#2,*215**437**141**797**89**56<CrLf>

The two records appear similar in that they both are terminated by a carriage return-linefeed and they both seem to contain multiple delimited data fields. A minor problem with the above two records is that they both contain a different number of data fields. Another complication is that they seem to contain both a comma and one or more asterisks as delimiter characters. In the first record it appears that we have four data fields: "Sample#1", "213", "32" and "23"
In the second record it appears that we have seven data fields:
"Sample#2", "215", "437", "141", "797", "89" and "56"

There are actually several ways to configure the Wedge to correctly parse both records in the above example. The most elegant method is to use the "Pre-Input Character Translation Table" to modify the delimiters so that they are all the same. For example, we could translate all asterisks to commas to end up with the two records shown below:

Sample#1,,213,,32,,,,23,,<CrLf>
Sample#2,,215,,437,,141,,797,,89,,56<CrLf>

Now both records obviously have multiple, delimited, data fields with a carriage return-linefeed signaling the end of each record. We now determine the maximum number of fields in a record by adding up the number of delimiters in each record and adding one. (i.e. the first record has 11 fields and the second record has 13 fields therefore the maximum number of data fields is 13).

Now, when you define the record structure in the Software Wedge you would select "Any Character Received" as the *Start Of Record Event* and "Carriage Return or CrLf Received" as the *End Of Record Event* and for the *Record Structure* you would choose "Multiple Delimited Data Fields" and specify that the maximum number of data fields per record is 13 with a comma delimiter separating each field.

When parsing records with multiple delimited data fields, if you choose a space character as your delimiter, then consecutive spaces would be treated as a single delimiter. Therefore, in the above example, if instead of translating asterisks to commas, you were to translate both commas and asterisks to *spaces*, and then you chose the space character as your delimiter, you would effectively end up with the two records shown below with each field now separated by a single (space) delimiter:

Sample#1 213 32 23 <CrLf>
Sample#2 215 437 141 797 89 56<CrLf>

Suppose you had a device that normally sent its output to a serial printer in the following manner with more than one line of data:

```
Device#        Height     Width    Length     <CrLf>
------------------------------------------------------- <CrLf>
    1            23         12       24       <CrLf>
    2            27         13       8        <CrLf>
    3            27         13       9        <CrLf>
    4            27         13       8        <CrLf>
------------------------------------------------------- <CrLf>
<FF>
```

(<CrLf> represents a carriage return-linefeed and <FF> represents the form feed character)

You can still think of the entire output shown above as a single record even though it contains several lines of data or you can think of each individual line as a complete record. (In fact, you could even think of each word or number outputted from the device as a complete, single field, data record).

Again there are hundreds of different ways to parse the entire output from the device using the different features available in the Software Wedge.

If you wanted to capture only the lines with numeric data and parse each of these lines into records with four fields, one way to accomplish this would be to specify the "Start Of Record Event" as "Numeric Character Received" and then specify the "End Of Record Event" as "Carriage Return or CrLf Received". Finally you would define the record structure as "Multiple Delimited Data Fields" with four fields per record and a space as the delimiter character. Because you chose "Numeric Character Received" as the "Start Of Record Event" and because there are no numeric characters in the first, second, seventh and eighth lines of data, these lines would be ignored.

If you wanted to capture the numeric data as above but with all 16 numbers in one record, you could again specify the "Start Of Record Event" as "Numeric Character Received" and then specify the "End Of Record Event" as "Special Character Received" and select the Dash ("-") character as the *special character* . Next, using the Pre-Input Character Translation Table, you would translate all carriage returns and linefeeds to spaces and finally you would define the record structure as "Multiple Delimited Data Fields" with 16 fields per record and the space character as the delimiter.

Once you understand all of the features provided in the Wedge, you can get very creative with how you deal with the data from your instrument(s).

The important points to remember are that the Wedge always waits until the selected "Start Of Record Event" occurs before it starts reading in any data after which it keeps reading in characters until the specified "End Of Record Event" occurs. When the "End Of Record Event" occurs, the Wedge takes the data record that it just received and parses it according to the definition of the Record Structure that you selected (i.e. Single Field, Multiple Delimited Fields or Multiple Fixed Length Fields). For each field that you have defined, the Wedge applies the chosen field Filter, Math Function and Format Expression. Finally the Wedge passes each field to the target application in sequence sending each field followed by the field's "Postamble" that you defined.

In many cases you could define the structure of your input data in many different ways. For example if your device transmitted the following data in bursts with some time between bursts, you could define the "End Of Record Event" as either "Carriage Return or CrLf Received" or as "Time Delay Between Records".

1, 23,12,24<CrLf>
2,27,13,8<CrLf>
3,27,13,9<CrLf>

If you specified "Carriage Return or CrLf Received" as the "End Of Record Event" then the Wedge would see the data as three records with four fields per record.

If you specified "Time Delay Between Records" and also used the "Pre-Input Translation Table" to translate carriage returns to commas and linefeeds to "Nul" or "Ignore", then the data would appear to the Wedge as follows:

1, 23,12,24,2,27,13,83,27,13,9

In other words you can think of the original data above as either three records containing four fields each or as a single record with 12 fields.

Again, as long as you understand the data from your serial device and have a clear understanding of the full capabilities of the Software Wedge, you can transform almost any serial data collection problem into an extremely simple task.

See Also: Defining The Input Data Record Structure
The Pre-Input Character Translation Table

More Software Wedge Configuration Examples

A common task for the Wedge is to interface a GagePort or GagePort NT to a PC and to take readings from a measuring tool like a gage or caliper directly into a spreadsheet, (typically Excel), and have each reading from the gage entered into the spreadsheet so that they are stacked up in a column. This section describes a typical configuration for both the GagePort and the Wedge.

Note: A GagePort transmits data records containing four comma delimited data fields with a carriage return at the end of the record. Many other devices transmit data in a similar manner therefore this example may be applicable to other devices. For example X-Rite Densitometers (used for measuring Cyan, Yellow, Magenta and Black color densities) also transmit records with four fields of data and a carriage return at the end of each record. The only difference between data from the densitometer and the data from a GagePort is that the densitometer uses a single space as the delimiter between each of the four data fields. It is very possible that whatever device you happen to be using also has a similar output to a GagePort therefore even if you are not using a GagePort, you may find this example to be very useful and educational.

Steps for setting up the GagePort:

1. Set up your GagePort in "Printer Mode" as per the instructions provided with the GagePort. This is done using the DIP switches on the GagePort. Also, use the DIP switches to set the communications parameters for the GagePort to 9600 baud, No Parity, Eight DataBits and One StopBit.

When set to Printer Mode, a reading is sent from the GagePort in either of the three following cases:

a. When you press the "transmit" button on your gage.
b. When you press the foot switch connected to the GagePort.
c. When a prompt is transmitted through the serial port to the GagePort requesting data.

2. Connect the GagePort to a COM port on your PC using the cable provided with the GagePort.

Steps for Setting up the Software Wedge:

For this situation, the easiest way to use the Wedge is to set it up in "Send Keystrokes" mode so that it will convert incoming serial data to keystrokes and thus cause your data from your GagePort to appear as if it is being typed in on your keyboard.

1. Run the Software Wedge by double clicking on its icon in either the Windows Program Manager or from the Start Menu in Windows 95.

2. From the Software Wedge MODE menu, select "Send Keystrokes To..." (even if it is already checked). This will cause a dialog box to appear asking for the "Title Bar Text" and "Command Line" for the application where you want the keystrokes sent. Clear out both of these items so that they are completely empty (no blank spaces or visible characters). By clearing out these items and thus not specifying an application for the Wedge to send the keystrokes to, the Wedge will simply act as a second keyboard and send the incoming serial data to whatever application has the current input focus, i.e. the current foreground program.

3. Select "Settings" from the PORT menu in the Wedge and when the Port Settings dialog box appears, choose the communications parameters: 9600 baud, No Parity, Eight DataBits and One StopBit and then click the OK button to return to the main menu.

4. To test that the GagePort is connected properly and that your serial cable is working, select "Analyze" from the Port menu in the Wedge. When the Port Analyze dialog box appears, transmit a reading to the Wedge by either pressing the Transmit button on your gage or by pressing the foot switch connected to the GagePort. If your gage does not have a transmit button or a foot switch, then place the cursor in the "Output Buffer" text box in the Analyze dialog box and type in an upper case "A" followed by a lower case "r" and then click the button marked "Send". If a reading is transmitted successfully from the GagePort, you should see some data in the text box marked "Input Buffer". You should receive 25 characters and the data should be in the following format:

NNNN,##########,UUUUU,PP<cr>

The meanings of the four fields are as follows:

NNNN	Four digit reading number
##########	Ten digit gage reading with decimal place and minus sign if the reading is negative.
UUUUU	Five blank spaces.
PP	Two digit port number.
<cr>	Trailing carriage return. (Should appear as a musical note in the Analyze window.)

If you do not receive any data in the Input Buffer in the Analyze window, then either the GagePort is not connected properly or it is not configured correctly. If you do get data in the Input Buffer but the data is either garbled or consists of unreadable characters, then you most likely have the wrong communications parameters selected in either the Wedge or in the GagePort. (The communications parameters set in both the Wedge and the GagePort must match.) Refer to the troubleshooting section of the Software Wedge users manual for further assistance. After you get good data in the Analyze window in the Software Wedge, click the Quit button to return to the main menu.

5. The next step is to configure the Wedge to parse and filter the data from the GagePort so that you get only the data that you are interested in to appear in your spreadsheet. In this case we will assume that you are only interested in the actual reading on the gage and that none of the other information transmitted by the GagePort is wanted.

Select "Input Data Record Structure" from the DEFINE menu in the Wedge. The first dialog to appear will prompt you for a "Start Of Record Event" and an "End of Record Event". For the Start of Record Event, choose "Any Character Received" and for the End of Record Event, choose "Carriage Return or CrLf Received" and click the CONTINUE button to proceed. In the next dialog box to appear prompting for a "Record Structure", select "Multiple Delimited Data Fields" and click the CONTINUE button. This will display another dialog box prompting for a "Delimiter Character" and the "Maximum Number of Data Fields". Choose the Comma (,) for the delimiter character and enter 4 for the maximum number of data fields and click the CONTINUE button to proceed.

The final dialog box to appear is the "Input Record Definition Editor" dialog box. This dialog box allows you to select filters that can be applied to each data field in the incoming serial data as well as define any additional keystrokes that you want issued before or after each individual data field is sent from the Wedge to the application where you want the data to go. Select "Ignore This Field" for the filter for Field 1 (leave all other text boxes blank) and click the button marked "Next Field". Select "Numeric Data Only" for the filter for Field 2 and place the cursor in the text box marked "Field Postamble Keystrokes" and click the button marked "Keystroke List". When the keystroke list appears, scroll the list until the word "ENTER" is highlighted and then click the OK button. This should cause the word "{ENTER}" to appear in the Field Postamble Keystrokes text box. Click the button marked "Next Field" and select "Ignore This Field" for the filter for Field 3 and click the "Next Field" button again and select "Ignore This Field" for the filter for Field 4. Finally, click the OK button to return to the main menu of the Wedge.

6. Select "Minimize on Activation" from the ACTIVATE menu and then save your work by selecting "Save" or Save As..." from the FILE menu. Save your configuration file using the file name GAGEPORT.SW3

7. The final step is to activate the Wedge by selecting "Test Mode" from the ACTIVATE menu. After the Wedge has been activated you can open up Excel and place the cursor in a cell and whenever you take a reading on your gage, the reading should appear in the cell and the cursor should automatically move down one cell.

A More Complex Example

This example explores some of the less obvious features of the Software Wedge and how to apply them to a more complex situation. In this example we will assume that you have already gone through the first four steps for setting up the Software wedge outlined in the previous example and have discovered that the data transmitted by your instrument is structured as follows with four lines of data and each line terminated by a carriage return:

Data Widget Coordinates<Cr>
x = 100.7, x deviation = .50<Cr>
y = -170.25, y deviation = .38<Cr>
z = 110, z deviation = -.5<Cr>

For our particular application, we are interested in capturing only the x, y and z values to a spreadsheet such that the three values appear next to each other on the same row but in separate columns. i.e. with one column of “x” values, one column of “y” values and one column of “z” values. We also want to ignore the first line of text and also each of the three “deviation” values.

If you think of the entire output from the device as a single data “Record” containing a group of data fields, the first thing we must do is to figure out a way to isolate the three x, y and z fields from the rest of the data using the parsing and filtering capabilities of the Software Wedge. There are actually several ways to achieve the results that we want using different features in the Software Wedge. One approach is to first use the “Pre Input Translation Table” to translate all commas to carriage returns so that our data appears to the Software Wedge as follows with the same delimiter character separating each individual data field:

Field 1	**Data Widget Coordinates<Cr>**
Field 2	**x = 100.7<Cr>**
Field 3	**x deviation = .50<Cr>**
Field 4	**y = -170.25<Cr>**
Field 5	**y deviation = .38<Cr>**
Field 6	**z = 110<Cr>**
Field 7	**z deviation = -.5<Cr>**

To perform this translation, select “Pre Input Translation Table” from the DEFINE menu and when the translation table appears, scroll down in the table until the “comma” character is hi-lighted and then click the button marked “Translate...”. In the ASCII chart that appears, scroll down to the carriage return character (ASCII 13) and click on the “OK” button in the ASCII chart dialog box. After you perform the translation, you can exit from the translation table by clicking the “OK” button in the translation table dialog box.

Looking at the data now, we can identify seven distinct data “fields”, each of which is delimited from the next by a carriage return.

The next step is to describe the structure of the data to the Software Wedge and also specify how the Wedge should parse and filter the data so that we get the results that we want. If we examine the data (with commas translated to carriage returns), we see that we would like to ignore the first, third, fifth and seventh fields and we also need to filter the second, fourth and sixth fields so that only the numeric data is sent to our spreadsheet. Finally, to get the data into our spreadsheet in three columns, we need to add a right arrow keypress after the second and fourth field and a down arrow and two left arrow keypresses after the sixth field.

To do this, select “Input Data Record Structure” from the DEFINE menu and for the “Start of Record Event” specify “Any Character Received”. For the “End of Record Event”, specify “Carriage Return or <CrLf> Received” and click on the “Continue” button to proceed. In the next dialog box, specify “Multiple Delimited Data Fields” as the record structure and click on the “Continue” button. The next dialog box to appear allows us to specify the delimiter character and also specify the maximum number of data fields in each data record. In this case our delimiter character is the carriage return (ASCII 13) and the maximum number of data fields in each record is seven. After entering these parameters, click on the “Continue“ button. The final dialog box to appear allows us to specify a filter for each field as well as any “Field Postamble Keystrokes” that we want to add to the data. Go through the following steps to specify the required filters and postamble keystrokes:

Place the cursor in the “Filter” list box and select “Ignore This Field” as the filter for Field 1. Use the “Next Field” and “Previous Field” buttons to scroll through the parameters for each of the seven data fields and select “Ignore This Field” as the filter for Fields 1, 3, 5 and 7 and select “Numeric Data Only” as the filter for Fields 2, 4 and 6.

Click the “Previous Field” button until you are back to the parameters for Field 2.

Place the cursor in the “Field Postamble Keystrokes” edit box and then click on the button marked “Keystroke List”. When the keystroke list dialog box appears, scroll down until the word “RIGHT” is hi-lighted and click the “OK” button in the keystroke list dialog. This should cause the word “**{RIGHT}**” to appear in the Field Postamble Keystrokes edit box.

Click on the “Next Field” button twice to advance to the parameters for Field 4 and repeat the previous step to add a right arrow keypress after Field 4.

Click on the “Next Field” button twice to advance to the parameters for Field 6 and enter the following text in the Field Postamble Keystrokes text box: **{DOWN}{LEFT 2}**
Click on the OK button to return to the main menu.

Finally, select “Test Mode” from the ACTIVATE menu, open up your spreadsheet and move the cursor to the cell where you want the first x value to appear and start collecting data.

See Also: Defining the Input Data Record Structure
Cool Wedge Tricks

Cool Wedge Tricks

Defining More Data Fields Than Are Actually Received
When you define the "Input Data Record Structure" in the Wedge, you first define the events that determine the start and the end of each data record. You then define how you want the Wedge to parse each data record by selecting a "Record Structure". You can have it parse your data records based on either the positions of delimiter characters within each record or by byte position in the record. The Wedge always stops reading in data when it encounters the specific "End Of Record Event" that you specify. If you define more data fields than are actually present in the data record then the remaining fields will be left empty. You could take advantage of additional empty data fields by using them to perform calculations on previous data fields using a "Math Expression" in the empty fields. For example suppose that you had a device that transmitted a pair of numbers, X1 and X2, in the following comma delimited format with a carriage return at the end of the data record:

X1,X2<Cr>

Suppose also that you wanted to pass these two numbers to another program along with the difference between the two numbers. To do this, you could define the input data record structure by specifying the End Of Record Event as "Carriage Return or CrLf Received" and then specifying the "Record Structure" as "Multiple Delimited Data Fields" with a comma as the delimiter and specifying *three* for the maximum number of data fields. The actual data has only two data fields before the carriage return so by specifying three for the maximum number of fields, you will end up with a third empty data field in the Wedge. For each field that you define in the Wedge, you can also define a Math Expression to be applied to the field. Math expressions in the Wedge can contain references to previous data fields by using the "Field(n)" expression. If you specified "Field(2) - Field(1)" for the math expression for Field(3) (the empty field), then you would end up with the difference between the two numbers X1 and X2 in Field(3) in the Wedge.

Defining Less Data Fields Than Are Actually Received
If you define your Input Data Record Structure with less data fields than actually appear in your data, the Wedge will simply ignore any additional data that it receives after the last field that you have defined in the Wedge. You could also take advantage of this behavior by in situations where a device transmits more data than you are actually interested in. For example suppose you had a device that transmitted 50 comma delimited data fields followed by a carriage return and you were only interested in the data that appears in the first three fields. You could define the input data record structure by specifying the End Of Record Event as "Carriage Return or CrLf Received" and then specifying the "Record Structure" as "Multiple Delimited Data Fields" with a comma as the delimiter and specifying *three* for the maximum number of data fields. The actual data has 50 data fields before the carriage return so by specifying three for the maximum number of fields, you will end up with only the first three data fields in the Wedge and the remaining 47 fields will be ignored.

Converting Binary Data to Decimal

If you have an instrument that transmits binary data and you would like to pass this data to another program as decimal data, you can use a combination of features available in the Wedge to perform this type of operation. Although the Wedge does not have a direct Binary To Decimal conversion function, it does allow you to select a "Binary To Hex" filter for each data field that you define and it also allows you to enter a "Hex to Decimal" function in the Math Expression for each field . If you apply the "Binary To Hex" filter to a data field, the Wedge will translate each ASCII character in the field to a two byte hexadecimal string. For example if you received an ASCII one followed by an ASCII 255 character, the Wedge would convert the two bytes to the four byte string "01FF". If you also entered the math expression "&h(Field(n))" for the field (where n is the number of the field containing the data), the Wedge would convert the hex data to decimal. In this above example, you would end up with the number 511 in the data field in the Wedge.

Dealing With Continuous Data

There are a few types of instruments that transmit data either continuously or in bursts with many data values arriving in succession over a short period of time. For example many older electronic balances were designed to transmit continuous data to a digital display so that you could watch the weight reading change on the display. They typically transmit up to 10 weight values per second and do not provide a way to stop the data from flowing. (Most newer balances are configurable to transmit data only in response to a prompt sent through the serial port or by pressing a "Print" button on the balance. This is the best way to use a balance with the Wedge.) There are also some devices like tensile strength testers or friction peel testers that perform a test over a short period of time and for the duration of the test they transmit a large number of data values.

Although you may not be able to configure the device to send less data, you may be able to configure the Wedge to ignore enough of the readings so that you are not overwhelmed with data. One way to do this is to configure the Wedge to treat multiple data records as one long record containing multiple fields and then simply ignore most of the fields. For example, most electronic balances simply transmit a weight reading followed by a carriage return and a line feed character. Normally you would define the "Input Data Record Structure" with "Carriage Return or CrLf Received" as the "End of Record Event" and then specify "Single Field Data Records" as the "Record Structure" for this type of data. If instead, you chose "Multiple Delimited Data Fields" as the "Record Structure" and chose the Carriage Return (ASCII 13) as the delimiter and also specified 30 for the "Maximum Number Of Data Fields" and finally chose "Ignore This Field" for the filters for Fields 1 through 29, you would effectively remove 29 out of each 30 data readings transmitted by the balance. If the balance sent ten readings per second, the Wedge would only pass a single reading through to another application once every three seconds.

Another way to control continuous data is to configure the Wedge so that it is initially suspended when you first activate it (select the "Activate Initially Suspended" option from the Activate menu). If the Wedge is Activated but suspended, it will continue to input serial data into its input buffer however it will not pass any data to another application until you resume it. Next, you could define one hot key that resets the Wedge and then define another hot key that resumes the Wedge for one data record. You can even assign both hot key actions to the same hot key keystroke so that both the reset and the enabling for one record is accomplished with the same keystroke. After you activate the Wedge it will not do anything until you press your hot key. When you press the hot key, the Wedge will reset itself (causing it to flush its input buffer) and then it will enable itself for one data record. After the next record is received, the Wedge will suspend itself again.

If you use the above technique, make sure that you select a "Start of Record Event" that will reliably determine the beginning of each data record (i.e. do not select "Any Character Received" as the Start of Record Event) . The reason for this is because if the device is in the middle of transmitting a data record when you press your hot key, the reset hot key action causes the Wedge to clear out its input buffer thus chopping off the first half of the data record being received. If you had "Any Character Received" selected as the Start of Record Event, the second half of the data record would be received as if it were a complete record. Choosing "Special Character Received" as the Start of Record Event and using the Line Feed (ASCII 10) as the "Special Character" would be a good choice in this situation. Since the device is transmitting a continual stream of data records with a carriage return and a line feed at the end of each data record, you could just as easily think of the line feed as the start of each record and the carriage return as the end. To remove the line feed and the carriage return from each record, you can use either the "Pre-Transfer Character Translation Table to translate them to "Nul" or you can apply a "Numeric Data Only" filter to the field(s) in the Wedge that contains them.

See Also: Defining the Input Data Record Structure
Special Considerations Regarding Delimiter Characters
Defining Hot Keys and Hot Key Actions
The Pre-Transfer Character Translation Table

Logging Data To A Disk File While Passing Data to Other Program Via DDE

Although the Wedge has an explicit "DDE Server Mode", it is actually a DDE Server no matter what mode it is in (i.e. "Send Keystrokes", "Log To Disk" or "DDE Server" modes). The reason that the Wedge has a DDE Server Mode at all is so that you can configure it to send a DDE command to another application after each input and thus force the other application to do something with the data that was just sent. Suppose that you wanted to log all data from the Wedge to a disk file *and* pass the data to another program via DDE in real time. You could set up the Wedge in Log To Disk mode and still pass the data through DDE links to another program. The only problem that you might run into is that you will not be able to configure the Wedge to send a DDE command to the other program and thus force it to do something like run a macro after each input data record. Many applications that support DDE also provide a way to monitor active DDE links so that the application automatically runs a macro when it detects new data from a DDE server. Excel for example has a function called "SetLinkOnData" that forces it to run a macro whenever data from a DDE server changes. Lotus 123 has a similar function called "DDE-Advise".

See Also: DDE Examples for Excel 5 & 7

Making Good Use Of The Pre-Input Character Translation Table

The "Pre-Input Character Translation Table" in the Wedge allows you to translate ASCII characters to other characters before being inputted to the Wedge. Typically, this feature is used to convert delimiter characters so that they will all be the same character when the Wedge receives the data. For example, suppose you had a device that transmitted the following data record:

123.53, 46.20, 03:45:21<CrLf>

In the above data record there are two different delimiter characters being used, the comma and the colon. If you translated the colons to commas in the "Pre-Input Character Translation Table", then the data record would appear to the Wedge as:

123.53, 46.20, 03,45,21<CrLf>

Now that all the delimiters are the same, you could define the record structure in the Wedge as "Multiple Delimited Data Fields" using the comma as the delimiter with five fields in each record.

The "Pre-Input Character Translation Table" also allows you to translate characters to either "Ignore" or "Void Record". The "Ignore translation simply removes the character from the input data as if the character did not exist. The "Void Record" translation causes the Wedge to disregard the entire current data record if a character that has been translated to "Void Record" appears anywhere within the record. This feature can be used reject certain data records from a device. For example, suppose you had a device that transmitted the following data:

Device#	Height	Width	Length	<CrLf>
----------	----------	----------	----------	<CrLf>
1	23	12	24	<CrLf>
2	27	13	8	<CrLf>
3	27	13	9	<CrLf>
4	27	13	8	<CrLf>
----------	----------	----------	----------	<CrLf>

Since you are probably not interested in the first line or the lines containing the dashes, you could use the "Void Record" translation to remove these lines by translating the dash character and the capitol "D" to "Void Record". After the translations, the Wedge would effectively receive the data as shown below:

1	23	12	24	<CrLf>
2	27	13	8	<CrLf>
3	27	13	9	<CrLf>
4	27	13	8	<CrLf>

See Also: The Pre-Input Character Translation Table

Sending keystrokes to a DOS Window with WinWedge
Although it is technically not possible for a Windows program to send keystrokes directly to a DOS application running in a DOS window, there is a way to cheat and get Windows to do the job for us. An interesting feature of Windows is that it supports the ability to "Cut and Paste" data between DOS and Windows programs using the Windows clipboard. In order to get the Wedge to send data to a DOS application in a DOS window you first have to send the data as keystrokes to a Windows program and then send additional keystrokes that first "cut" the data from the Windows application, switch the focus to the DOS window and finally "paste" the data into the DOS window.

To try this technique, configure the Wedge in "Send Keystrokes" mode specifying the NotePad program as the target for all keystrokes from the Wedge. Next, configure the Wedge to work with your serial device and then add the following characters to the end of the "Field Postamble Keystrokes" for the very last data field that you have defined in the Wedge: **%ea^x%{TAB}%{ }ep**

Finally, open a DOS session in a window (not full screen) and transmit some data from your device. The data should first pop into the NotePad program and then be cut from NotePad and pasted into the DOS window.

The keystrokes: "%ea" mean "Alt+e a" which causes the NotePad program to select all the text in the NotePad Window. The keystrokes: "^x" mean "Ctrl+x" which causes NotePad to cut the currently selected text to the clipboard. The keystrokes: "%{TAB}" mean "Alt+Tab" which causes Windows to switch the focus to the last window that was active before the NotePad program received the focus (i.e. the DOS window). Finally the keystrokes: "%{ }ep" mean "Alt+space e p" which opens the control box in the DOS window and selects "Edit" and "Paste".

The next time you transmit a data record from your device, the Wedge will switch the focus back to the NotePad application automatically and do the same thing all over again. If you have a reasonably fast PC you will barely notice the NotePad program popping open in front of your DOS window.

Note: A true DOS version of the Software Wedge is available that may be better suited for passing data to a DOS application program. The Software Wedge for DOS is a RAM resident (TSR) program that passes serial data directly to other DOS programs by stuffing the DOS keyboard buffer. Although it is a DOS application, it runs extremely well in Windows inside a DOS Window and is an excellent tool for passing serial data to other DOS based programs.

Sending data as keystrokes to more than one application at a time

In the previous example we showed how to send data from the Wedge to a DOS application by first sending the data as keystrokes to the NotePad program and then cutting the data out of NotePad, switching the focus to a DOS window and pasting the data in. You can use the same technique to have the Wedge send data to two (or more) Windows or DOS programs at a time. For example, if in the previous example we were to copy the data from NotePad instead of cutting the data, we would end up with both the NotePad and the DOS program receiving the data.

All you need to do to pass data to two Windows programs is to send the data to the first application and then use whatever keystrokes are necessary to copy the data to the clipboard, switch the focus to the second application and finally paste the data in. Even if neither of the applications provides an explicit copy or paste function, you can always copy text to the clipboard in any Windows program by first selecting the text and issuing the keystroke Ctrl+Insert. In a "Field Postamble Keystroke" macro in the Wedge, Ctrl+Insert is represented by: ^{INSERT}. Likewise, you can paste text data from the clipboard into any Windows program using the keystrokes Shift+Insert. In the Wedge Shift+Insert is represented by: +{INSERT}. To select text in any program you can hold down the shift key while using the arrow keys, Home, End, Page Up and Page Down keys to highlight and select characters. Windows will also switch the focus between all running application when you hold the Alt key down and press the Tab key. Windows maintains a list of all currently running applications with the application that currently has the input focus placed at the top of the list. If you switch to another application, the new application moves to the top of the list and the previously active application moves down one spot in the list. When hold the ALT key down, each time you press the tab key, Windows moves down the list one item and selects the application at that position in the list. When you release the Alt key, the selected application is given the focus and is moved to the top of the list pushing all other application that were above it in the list down one spot.

For example, if you had the Wedge set up to send keystrokes to NotePad and you also wanted to send the data to the Windows Write (WordPad in Windows 95) you could use the following procedure:

Set up the Wedge in "Send Keystrokes" mode specifying the NotePad program as the target for all keystrokes from the Wedge. Next, configure the Wedge to work with your serial device and then add the following characters to the end of the "Field Postamble Keystrokes" for the very last data field that you have defined in the Wedge: **({LEFT}{HOME})^{INSERT}{DOWN}%{TAB}+{INSERT}**

Activate the Wedge and then launch the Write program followed by the NotePad program. At this point NotePad has the focus and is therefore at the top of the Window list with the Write program directly underneath it. When you input data from your device, it will be sent to both NotePad and Write. The keystrokes: +({LEFT}{HOME}) means hold the shift key while pressing the left arrow and the home key. This selects all text in the current line. The keystrokes: ^{INSERT} means Ctrl+Insert and causes all selected text to be copied to the clipboard. The {DOWN} keystroke that follows is there to simply de-select the currently selected text. The keystrokes: %{TAB} means Alt-Tab and causes Windows to switch the focus to the Write program. Finally, the keystrokes: +{INSERT} means shift+insert and performs the job of pasting the data from the clipboard into the Write program.

To send the data to three Windows programs at a time you could use the following "Field Postamble Keystrokes": **+({LEFT}{HOME})^{INSERT}{DOWN}%{TAB}+{INSERT}%{TAB 2}+{INSERT}**

Using The Wedge in combination with the Recorder application under with Windows 3.1
Windows 3.x is shipped with an extremely useful utility called "Recorder" that allows you to record sequences of keystrokes, mouse moves and mouse clicks. You can then assign your recorded sequences to hot keys so that all you need to do to play back a long sequence is press a single hot key. The Recorder program can be found in your Accessories program group in Window 3.1. Recorder is probably the single most time saving utility on your PC but unfortunately it is also the least understood and least used accessory. In fact, Microsoft does not even provide Recorder as part of Windows 95 because they found that hardly anyone was using it in Windows 3.1. (Unfortunately, the 16 bit version of Recorder that is shipped with Windows 3.1 does not work in Windows 95 or NT so if you are running Windows 95 or NT this discussion will be of little value.) Because the Recorder program allows you to assign macros to a single hot key keystrokes, you could invoke a Recorder macro from the Wedge by placing the hot key keystroke for a Recorder macro in either the "Record Preamble Keystrokes" or the "Field Postamble Keystrokes" when you define your "Input Data Record Structure" in the Wedge. You could also use the "Pre-Transfer Character Translation Table" to translate specific characters that might appear in your serial data to keystrokes that invoke different Recorder macros. Using the Wedge in combination with the Recorder thus allows you to do practically anything you can imagine with the data coming in the serial port. For example you could set up the Wedge and Recorder so that whenever a specific character was received through the serial port, an entire sequence of operations took place including opening and closing applications, deleting files, etc.. If you can do it manually then Recorder can automate the job for you with a macro and the Wedge can send the necessary hot keys to invoke the macro.

See Also:
Specifying Pre / Postamble Keystrokes (Send Keystrokes Mode)
Pre Transfer Character Translation Table

Invoking hot keys in other applications
Almost all spreadsheets, databases, word processors and most other large scale application programs allow you to either record macros or create subroutines that can be invoked with a hot key while you are working within the application. For example when you write a subroutine in Excel, you can assign the subroutine to a "Shortcut Key" so that all you have to do to run the subroutine is press the shortcut key keystroke. Similar to the previous discussion about using the Wedge to invoke Recorder macros, you could invoke Excel subroutines from the Wedge by placing the shortcut key keystroke for a subroutine in either the "Record Preamble Keystrokes" or the "Field Postamble Keystrokes" when you define your "Input Data Record Structure" in the Wedge. You could also use the "Pre-Transfer Character Translation Table" to translate specific characters that might appear in your serial data to keystrokes that would invoke different Excel subroutines.

Using the Wedge to update an HTML file to provide real-time data for an internet home page
Suppose that you had an instrument that provided some sort of data on a regular basis and you wanted to update an HTML file with new data each time it was transmitted from the instrument. One way to do this is to set up the Wedge in "Send Keystrokes" mode and have it send keystrokes containing HTML code along with the data from the instrument to the NotePad program.

In this case we first set up the Wedge so that each time it receives new data from the instrument it opens the NotePad program and automatically loads in an existing HTML document. We also define the "Record Preamble Keystrokes" in the Wedge that will select all the text in the NotePad window and overwrite it with all the HTML code that will go before the actual data from the instrument. Finally we define the "Field Postamble Keystrokes" after the last data field defined in the Wedge to contain all the HTML code that will go after the actual data. We also need to add in a few additional keystrokes that will save the new HTML file overwriting the old one and finally close the NotePad program.

When you select "Send Keystrokes" from the mode menu in the Wedge, it will ask you for the "Application Title Bar Text" and the "Command Line" for the application that you want to receive the keystrokes. In this case you would enter: **NotePad** for the Application Title Bar Text and then specify: **Notepad.Exe mypage.htm** where "mypage.htm" is the full path and file name for the HTML file that you want to update. When you specify a file name on the command line used to launch NotePad, NotePad will automatically load in the specified file.

The following HTML code is a minimal example of what you might need in your HTML document. (The META command on the third line will cause most browsers to automatically refresh the page every 5 seconds. Please refer to your favorite HTML code reference for an explanation of the HTML tags in the following code.)

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Refresh" CONTENT=5>
<TITLE>Real Time Data Provided By Software Wedge</TITLE>
</HEAD>
<BODY>
<H1>Serial Data goes here maybe</H1>
</BODY>
</HTML>
```

To get the Wedge to write the above code you would use the following text in the "Record Preamble Keystrokes" in the Wedge:

%ea<HTML>~<HEAD>~<META HTTP-EQUIV="Refresh" CONTENT=5>~<TITLE>Real Time Data Provided By Software Wedge</TITLE>~</HEAD>~<BODY>~<H1>

The keystrokes %ea means "Alt+e a" and will cause Notepad to select all the text in the Notepad window. The tilde (~) characters cause the Wedge to send "Enter" keystrokes. The tilde is equivalent in meaning to the word "{ENTER}" when defining preamble and postamble keystrokes in the Wedge.

The "Field Postamble Keystrokes" after the last data field defined in the Wedge would be:

</H1>~</BODY>~</HTML>~%fx~

The keystrokes %fx mean "Alt+f x" and when sent to Notepad, cause it to select "Exit" from its File menu. The final tilde (Enter keystroke) is required because Notepad will prompt you with a message box asking you if you want to save changes to the file. If the Enter key is pressed in response to the prompt, Notepad overwrites the existing file with the new data.

Each time that the Wedge receives new data from the device, it will launch Notepad automatically loading in the old HTML file and then it overwrites the file with the same HTML code but with new data from the instrument. Finally it unloads Notepad and saves the new file and overwriting the old one. The process will repeat itself each time you receive new data from the instrument.

FileWedge

FileWedge is almost identical to the Software Wedge except that instead of taking input from a serial port, FileWedge reads its input from a disk file. Like the Software Wedge, the concept of FileWedge is extremely powerful because it allows you to take data from any disk file and pump the data directly into any Windows application either by converting the data to keystrokes or by passing the data through DDE links. Like the Software Wedge, FileWedge also has the ability to parse, filter, format and translate data and also to write its output to a disk file instead of another application program. This means that you could even use FileWedge to convert data from one file format to another.

If you understand how to use the Software Wedge then you already know how to use FileWedge.

There are only a few minor differences between FileWedge and the Software Wedge as discussed below.

Differences Between FileWedge and The Software Wedge

Since FileWedge reads input from a disk file and not from a serial port, instead of having a "PORT" sub menu in the main menu, FileWedge has a "INPUT" sub menu. Also, the "INPUT" sub menu contains the option "Input File Name" in place of a "Port Settings" option. Selecting the "Source File Name" option opens a dialog box where you can select the name of the file containing the data that you want to read into another program or disk file. Also, when you select the "Analyze" option in the “Input” menu, FileWedge will display a window where you can examine the contents of the source data file.

Since FileWedge does not have anything to do with serial ports, the "Output Strings" option and the “Output Buffer DDE Link” option is not available in the "Define" sub menu. Also, the only hot key actions that you can define for File Wedge are: "Reset FWedge", "Disable FWedge", "Enable FWedge" and "Enable 1 Record".

Another difference between FileWedge and the Software Wedge is that when you define the structure of your input data, you will not be able to specify "Time Delay Between Records" as the “End of Record Event”. Since data is being read from a disk file and not asynchronously through a serial port, this option would be totally meaningless. FileWedge also does not allow you to specify Math Expressions to be applied to data fields.

To communicate with FileWedge using Dynamic Data Exchange (DDE), use the DDE Aplication Name **FileWedg** and set the DDE Topic to the name of the input data file that you configured File Wedge to use as the input data source. The DDE Item names available in File Wedge are the same as in the Software Wedge (i.e. **Field(1)**, **Field(2)**, etc., **RecordNumber** and **DateTime**)

One feature that FileWedge has that the Software Wedge does not is an option in the DEFINE menu that allows you to specify a “Time Delay Between Records”. Do not confuse this with the "Time Delay Between Records" “End of Record Event” option in the Software Wedge. This option is used to slow down File Wedge by allowing you to specify a time delay that FileWedge should wait after each data record from your input file is transferred to a target application.

Because FileWedge reads its input from a disk file, it does not have to wait for data from a serial device; the source data already exists in a disk file. When FileWedge is activated in "Input As Keystrokes" mode, it will start sending keystrokes immediately, causing data to stream into the target application immediately after you activate it. This is the reason for the “Time Delay Between Records” feature in FileWedge. Because of this behavior, you may also want to install FileWedge initially disabled and then define a hot key that you can use to enable it after you have had enough time to start another application program and get to the point where you want data to be entered.

You may also want to define a hot key that you can use to disable FileWedge if things do not go as you expect. FileWedge will not stop sending keystrokes until the entire file has been read in. If you specify a very large input file and the data is not flowing into your application program the way you want it to, you will most certainly want to have a quick and easy way to turn FileWedge off.

Finally, FileWedge also has an option "Exit When Finished" in its "Activate" menu that you can select to instruct FileWedge to automatically unload itself from memory after it reaches the end of the input data file.

The original idea behind FileWedge was to compliment the Software Wedge. Because the Software Wedge allows you to log serial data directly to a disk file, you could use the Software Wedge to log serial data from an instrument directly to a disk file and then use FileWedge to send the data to an application program at the end of the day. Another use for FileWedge is to force data from any disk file into any Windows application program. For example, suppose you have a file containing a list of names and addresses and you need to get this data into an application program that does not have the ability to import the data. You could use FileWedge to “type” the data into the application for you. Because FileWedge can also write its output to a disk file, you could even use FileWedge to convert data from one file format to another. For example, suppose you have a file that contains data in a fixed length record format and you want to import this data into a database program that can only import comma delimited data records. You could use FileWedge to read in the source file, parse each data record into its component data fields and then write the data back out to a new disk file with comma delimiters between each data field.

FileWedge is an extremely powerful tool. You will almost certainly discover a uniquely problematic situation where FileWedge proves to be the perfect solution. For more detailed information on using FileWedge, refer to the on-line help available by selecting “Index” from the FileWedge “Help” menu.

TCP/Wedge (WinWedge Pro 32 Bit Edition Only)

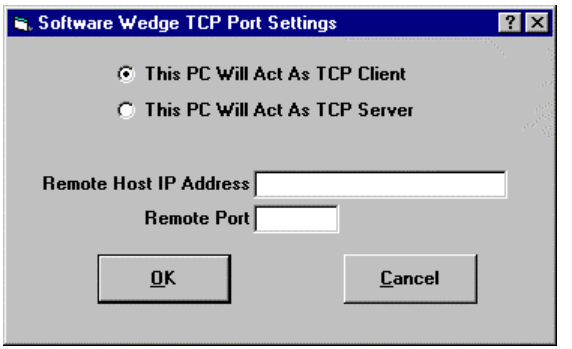
TCP/Wedge is also almost identical to the Software Wedge except that instead of providing an interface to a serial port, TCP/Wedge communicates through a TCP/IP network port address. TCP/Wedge allows you to take data from any TCP/IP network data source and feed that data directly into any Windows application. Also, like Software Wedge, TCP/Wedge works by either converting the data to keystrokes or by passing the data through DDE links. TCP/Wedge also has the ability to parse, filter, format and translate data and also to write data to a disk file instead of another program. TCP/Wedge provides full two way I/O capabilities allowing you to send and receive data to any device connected over a TCP/IP network, including an Intranet or the Internet!

Note: TCP/Wedge is provided with the 32 bit version of the Software Wedge only.
To use TCP/Wedge, you must have the TCP/IP network protocol installed in your PC. You can install the TCP/IP network protocol by selecting the Network icon in the Window control panel.

There are only a few differences between TCP/Wedge and the Software Wedge thus you can use this users manual for both products. The differences between the two products is discussed below.

Differences Between TCP/Wedge and The Software Wedge

Since TCP/Wedge communicates through a TCP/IP port address and not through a serial port, instead of allowing you to select any serial communications parameters, TCP/Wedge allows you to specify an IP address and port number. When you select "Settings" from the PORT menu in TCP/Wedge, the following dialog box will appear:



TCP/Wedge can operate as a TCP Client or as a TCP server. If you choose the option to set up TCP/Wedge as a TCP Client, then you will need to enter the IP address ("Remote Host IP Address") and the Port number ("Remote Port") of the TCP Server that you want TCP/Wedge to connect to. When you set up TCP/Wedge as the client, TCP/Wedge will be responsible for establishing the connection to the Remote IP Address and Port number that you specify. The connection to the remote IP Address will occur when you activate TCP/Wedge by selecting either "Test Mode" or "Normal Mode" from the ACTIVATE menu or when you open the "Analyze" window by selecting "Analyze" from the PORT menu.

If you choose to set up TCP/Wedge as a TCP Server, then TCP/Wedge will use the IP address of the computer that you are using as the "Local IP Address" however you will need to specify a "Local Port" number that you would like to use. In this case, the remote client will be responsible for initiating the connection to TCP/Wedge using the Local IP address for your PC and the Local Port number that you specify. When you select the option "This PC Will Act as TCP Server" in the TCP Wedge Port Settings dialog box, the Local IP Address for your computer will be displayed in gray text in IP Address text box. If your PC does not have the TCP/IP network protocol installed or if your PC does not have an IP Address assigned to it, you will not be able to select the option to set up TCP/Wedge as a TCP Server. If you set up TCP/Wedge as a TCP Server, after you activate TCP/Wedge in either Test or Normal mode, it will simply wait for another computer on the network to establish a TCP/IP connection to it using the Local IP Address of the PC where TCP/Wedge is running and the Port Number that you specified.

TCP/Wedge is functionally identical to the Software Wedge except for a few minor differences. Since TCP/Wedge does not communicate over a serial port you will not be able to select any options that are specific to a serial port. For example, when you define serial output strings or hot keys and hot key actions in TCP/Wedge you will not find any options or hot key actions that relate specifically to serial ports. This includes the ability to set the states of any serial port hardware handshaking lines like RTS and DTR. Otherwise, TCP/Wedge is identical to Software Wedge.

To communicate with TCP/Wedge using Dynamic Data Exchange (DDE), use the DDE Aplication Name **TCPWedge** and set the DDE Topic to the IP Address that you entered in the TCPWedge Port Settings dialog box. If TCP/Wedge is set up to be a Server, the DDE Topic would be the IP Address of the PC that TCP/Wedge is running on. The DDE Item names available in TCP/Wedge are the same as in the Software Wedge (i.e. **Field(1)**, **Field(2)**, etc., **RecordNumber** and **DateTime**)

There are a number interesting ways that you could use TCP/Wedge including sending and receiving data over an Intranet or the Internet directly from within any Windows program like Excel or Access. Several network hardware companies also sell protocol converters that convert from RS232 to an Ethernet connection. Using one of these devices, you could connect an instrument with an RS232 output directly to a TCP/IP network. You could then run TCP/Wedge in any PC on the network to communicate with the serial device. Normally, to accomplish the same functionality, you would have to install a PC and a network connection near the device and then plug the device into the serial port on the PC, use the Software Wedge to pull in data from the device, and finally, use either NetDDE or some other program like TelNET to pass the data on to another workstation on the network. Using an RS232 to TCP/IP converter, you would not need to install a complete networked PC at the location where the device is at. You could simply install a network connection and the RS232 to TCP/IP converter and then use TCP/Wedge on any other PC in the network to communicate with the device. One company that makes a good RS232 to Ethernet converter for around $500 (model# MSS100) is LanTronix , Tel: 714-453-3990 (http://www.lantronix.com).

INTRODUCTION TO SERIAL COMMUNICATIONS

All IBM PC and compatible computers are typically equipped with two serial ports and one parallel port. Although these two types of ports are used for communicating with external devices, they work in different ways.

A parallel port sends and receives data eight bits at a time over 8 separate wires. This allows data to be transferred very quickly; however, the cable required is more bulky because of the number of individual wires it must contain. Parallel ports are typically used to connect a PC to a printer and are rarely used for much else. A serial port sends and receives data one bit at a time over one wire. While it takes eight times as long to transfer each byte of data this way, only a few wires are required. In fact, two-way (full duplex) communications is possible with only three separate wires - one to send, one to receive, and a common signal ground wire.

Bi-directional Communications

The serial port on your PC is a full-duplex device meaning that it can send and receive data at the same time. In order to be able to do this, it uses separate lines for transmitting and receiving data. Some types of serial devices support only one-way communications and therefore use only two wires in the cable - the transmit line and the signal ground.

Synchronous And Asynchronous Communications

There are two basic types of serial communications, synchronous and asynchronous. With synchronous communications, the two devices initially synchronize themselves to each other, and then continually send characters to stay in sync. Even when data is not really being sent, a constant flow of bits allows each device to know where the other is at any given time. That is, each character that is sent is either actual data or an idle character. Synchronous communications allows faster data transfer rates than asynchronous methods, because additional bits to mark the beginning and end of each data byte are not required. The serial ports on IBM-style PCs are asynchronous devices and therefore only support asynchronous serial communications.

Asynchronous means "no synchronization", and thus does not require sending and receiving idle characters. However, the beginning and end of each byte of data must be identified by start and stop bits. The start bit indicate when the data byte is about to begin and the stop bit signals when it ends. The requirement to send these additional two bits cause asynchronous communications to be slightly slower than synchronous however it has the advantage that the processor does not have to deal with the additional idle characters.

An asynchronous line that is idle is identified with a value of 1, (also called a mark state). By using this value to indicate that no data is currently being sent, the devices are able to distinguish between an idle state and a disconnected line. When a character is about to be transmitted, a start bit is sent. A start bit has a value of 0, (also called a space state). Thus, when the line switches from a value of 1 to a value of 0, the receiver is alerted that a data character is about to come down the line.

Communicating By Bits

Once the start bit has been sent, the transmitter sends the actual data bits. There may either be 5, 6, 7, or 8 data bits, depending on the number you have selected. Both receiver and the transmitter must agree on the number of data bits, as well as the baud rate. Almost all devices transmit data using either 7 or 8 databits.

Notice that when only 7 data bits are employed, you cannot send ASCII values greater than 127. Likewise, using 5 bits limits the highest possible value to 31. After the data has been transmitted, a stop bit is sent. A stop bit has a value of 1 - or a mark state - and it can be detected correctly even if the previous data bit also had a value of 1. This is accomplished by the stop bit's duration. Stop bits can be 1, 1.5, or 2 bit periods in length.

The Parity Bit

Besides the synchronization provided by the use of start and stop bits, an additional bit called a parity bit may optionally be transmitted along with the data. A parity bit affords a small amount of error checking, to help detect data corruption that might occur during transmission. You can choose either even parity, odd parity, mark parity, space parity or none at all. When even or odd parity is being used, the number of marks (logical 1 bits) in each data byte are counted, and a single bit is transmitted following the data bits to indicate whether the number of 1 bits just sent is even or odd.

For example, when even parity is chosen, the parity bit is transmitted with a value of 0 if the number of preceding marks is an even number. For the binary value of 0110 0011 the parity bit would be 0. If even parity were in effect and the binary number 1101 0110 were sent, then the parity bit would be 1. Odd parity is just the opposite, and the parity bit is 0 when the number of mark bits in the preceding word is an odd number. Parity error checking is very rudimentary. While it will tell you if there is a single bit error in the character, it doesn't show which bit was received in error. Also, if an even number of bits are in error then the parity bit would not reflect any error at all.

Mark parity means that the parity bit is always set to the mark signal condition and likewise space parity always sends the parity bit in the space signal condition. Since these two parity options serve no useful purpose whatsoever, they are almost never used.

Baud Versus Bits Per Second

The baud unit is named after Jean Maurice Emile Baudot, who was an officer in the French Telegraph Service. He is credited with devising the first uniform-length 5-bit code for characters of the alphabet in the late 19th century. What baud really refers to is modulation rate or the number of times per second that a line changes state. This is not always the same as bits per second (BPS). If you connect two serial devices together using direct cables then baud and BPS are in fact the same. Thus, if you are running at 19200 BPS, then the line is also changing states 19200 times per second. But when considering modems, this isn't the case.

Because modems transfer signals over a telephone line, the baud rate is actually limited to a maximum of 2400 baud. This is a physical restriction of the lines provided by the phone company. The increased data throughput achieved with 9600 or higher baud modems is accomplished by using sophisticated phase modulation, and data compression techniques.

RS-232C

RS-232 stands for Recommend Standard number 232 and C is the latest revision of the standard. The serial ports on most computers use a subset of the RS-232C standard. The full RS-232C standard specifies a 25-pin "D" connector of which 22 pins are used. Most of these pins are not needed for normal PC communications, and indeed, most new PCs are equipped with male D type connectors having only 9 pins.

DCE And DTE Devices

Two terms you should be familiar with are DTE and DCE. DTE stands for Data Terminal equipment, and DCE stands for Data Communications Equipment. These terms are used to indicate the pin-out for the connectors on a device and the direction of the signals on the pins. Your computer is a DTE device, while most other devices are usually DCE devices.
If you have trouble keeping the two straight then replace the term "DTE device" with "your PC" and the term "DCE device" with "remote device" in the following discussion.

The RS-232 standard states that DTE devices use a 25-pin male connector, and DCE devices use a 25-pin female connector. You can therefore connect a DTE device to a DCE using a straight pin-for-pin connection. However, to connect two like devices, you must instead use a null modem cable. Null modem cables cross the transmit and receive lines in the cable, and are discussed later in this chapter. The listing below shows the connections and signal directions for both 25 and 9-pin connectors.

25 Pin Connector on a DTE device (PC connection)

Pin	Name	Direction of signal
1	Protective Ground	
2	Transmitted Data (TD)	Outgoing Data (from a DTE to a DCE)
3	Received Data (RD)	Incoming Data (from a DCE to a DTE)
4	Request To Send (RTS)	Outgoing flow control signal controlled by DTE
5	Clear To Send (CTS)	Incoming flow control signal controlled by DCE
6	Data Set Ready (DSR)	Incoming handshaking signal controlled by DCE
7	Signal Ground	Common reference voltage
8	Carrier Detect (CD)	Incoming signal from a modem
20	Data Terminal Ready (DTR)	Outgoing handshaking signal controlled by DTE
22	Ring Indicator (RI)	Incoming signal from a modem

9 Pin Connector on a DTE device (PC connection)

Pin	Name	Direction of signal
1	Carrier Detect (CD) (from DCE)	Incoming signal from a modem
2	Received Data (RD)	Incoming Data from a DCE
3	Transmitted Data (TD)	Outgoing Data to a DCE
4	Data Terminal Ready (DTR)	Outgoing handshaking signal
5	Signal Ground	Common reference voltage
6	Data Set Ready (DSR)	Incoming handshaking signal
7	Request To Send (RTS)	Outgoing flow control signal
8	Clear To Send (CTS)	Incoming flow control signal
9	Ring Indicator (RI) (from DCE)	Incoming signal from a modem

The TD (transmit data) wire is the one through which data from a DTE device is transmitted to a DCE device. This name can be deceiving, because this wire is used by a DCE device to receive its data. The TD line is kept in a mark condition by the DTE device when it is idle. The RD (receive data) wire is the one on which data is received by a DTE device, and the DCE device keeps this line in a mark condition when idle.

RTS stands for Request To Send. This line and the CTS line are used when "hardware flow control" is enabled in both the DTE and DCE devices. The DTE device puts this line in a mark condition to tell the remote device that it is ready and able to receive data. If the DTE device is not able to receive data (typically because its receive buffer is almost full), it will put this line in the space condition as a signal to the DCE to stop sending data. When the DTE device is ready to receive more data (i.e. after data has been removed from its receive buffer), it will place this line back in the mark condition. The complement of the RTS wire is CTS, which stands for Clear To Send. The DCE device puts this line in a mark condition to tell the DTE device that it is ready to receive the data. Likewise, if the DCE device is unable to receive data, it will place this line in the space condition. Together, these two lines make up what is called RTS/CTS or "hardware" flow control. The Software Wedge supports this type of flow control, as well as Xon/XOff or "software" flow control. Software flow control uses special control characters transmitted from one device to another to tell the other device to stop or start sending data. With software flow control the RTS and CTS lines are not used.

DTR stands for Data Terminal Ready. Its intended function is very similar to the RTS line. DSR (Data Set Ready) is the companion to DTR in the same way that CTS is to RTS. Some serial devices use DTR and DSR as signals to simply confirm that a device is connected and is turned on. The Software Wedge sets DTR to the mark state when the serial port is opened and leaves it in that state until the port is closed. The DTR and DSR lines were originally designed to provide an alternate method of hardware handshaking. It would be pointless to use both RTS/CTS and DTR/DSR for flow control signals at the same time. Because of this, DTR and DSR are rarely used for flow control.

CD stands for Carrier Detect. Carrier Detect is used by a modem to signal that it has a made a connection with another modem, or has detected a carrier tone.

The last remaining line is RI or Ring Indicator. A modem toggles the state of this line when an incoming call rings your phone.

The Carrier Detect (CD) and the Ring Indicator (RI) lines are only available in connections to a modem. Because most modems transmit status information to a PC when either a carrier signal is detected (i.e. when a connection is made to another modem) or when the line is ringing, these two lines are rarely used.

Cable Lengths

The RS-232C standard imposes a cable length limit of 50 feet. You can usually ignore this "standard", since a cable can be as long as 10000 feet at baud rates up to 19200 if you use a high quality, well shielded cable. The external environment has a large effect on lengths for unshielded cables. In electrically noisy environments, even very short cables can pick up stray signals. The following chart offers some reasonable guidelines for 24 gauge wire under typical conditions. You can greatly extend the cable length by using additional devices like optical isolators and signal boosters. Optical isolators use LEDs and Photo Diodes to isolate each line in a serial cable including the signal ground. Any electrical noise affects all lines in the optically isolated cable equally - including the signal ground line. This causes the voltages on the signal lines relative to the signal ground line to reflect the true voltage of the signal and thus canceling out the effect of any noise signals.

Baud Rate	Shielded Cable Length	Unshielded Cable Length
110	5000 (feet)	1000 (feet)
300	4000	1000
1200	3000	500
2400	2000	500
4800	500	250
9600	250	100

Cables, Null Modems, And Gender Changers

In a perfect world, all serial ports on every computer would be DTE devices with 25-pin male "D" connectors. All other devices to would be DCE devices with 25-pin female connectors. This would allow you to use a cable in which each pin on one end of the cable is connected to the same pin on the other end. Unfortunately, we don't live in a perfect world. Serial ports use both 9 and 25 pins, many devices can be configured as either DTE or DCE, and - as in the case of many data collection devices - may use completely non standard or proprietary pin-outs. Because of this lack of standardization, special cables called null modem cables, gender changers and custom made cables are often required.

Null Modem Cables and Null Modem Adapters

If you connect two DTE devices (or two DCE devices) using a straight RS232 cable, then the transmit line on each device will be connected to the transmit line on the other device and the receive lines will likewise be connected to each other. A Null Modem cable or Null Modem adapter simply crosses the receive and transmit lines so that transmit on one end is connected to receive on the other end and vice versa. In addition to transmit and receive, DTR & DSR, as well as RTS & CTS are also crossed in a Null modem connection.
Null modem adapter are available at most computer and office supply stores for under $5.

9 Pin To 25 Pin Adapters

The following table shows the connections inside a standard 9 pin to 25 pin adapter.

9-Pin Connector		25 Pin Connector	
Pin 1	DCD	Pin 8	DCD
Pin 2	RD	Pin 3	RD
Pin 3	TD	Pin 2	TD
Pin 4	DTR	Pin 20	DTR
Pin 5	GND	Pin 7	GND
Pin 6	DSR	Pin 6	DSR
Pin 7	RTS	Pin 4	RTS
Pin 8	CTS	Pin 5	CTS
Pin 9	RI	Pin 22	RI

Gender Changers

The final problem you may encounter is having two connectors of the same gender that must be connected. Again, you can purchase gender changers at any computer or office supply store for under $5.

Note: The parallel port on a PC uses a 25 pin female connector which sometimes causes confusion because it looks just like a serial port except that it has the wrong gender. Both 9 and 25 pin serial ports on a PC will always have a male connector.